UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

DESARROLLO DE UNA EXTENSIÓN A GEONODE BAJO EL FRAMEWORK DJANGO PARA IMPLEMENTAR UN ATLAS DIGITAL EN LINEA

REPORTE DE ACTIVIDAD DE APOYO A LA INVESTIGACION

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A :

VÍCTOR MAYA HIGUERA

TUTOR:
M. EN C. FRANCISCO JAVIER OSORNO COVARRUBIAS

2016

1. Datos del alumno
    - Maya Higuera Víctor
    - 22 11 53 21
    - Universidad Nacional Autónoma de México
    - Facultad de Ciencias
    - Ciencias de la Computación
    - 306192342
2. Datos del tutor
    - M. en C. Francisco Javier Osorno Covarrubias
3. Datos del sinodal 1
    - Dr. Stéphane Robert André Couturier
4. Datos del sinodal 2
    - M. en C. Josafat Isai Guerrero Iñiguez
5. Datos del sinodal 3
    - Dr. Gustavo de la Cruz Martínez
6. Datos del sinodal 4
    - Dr. Miguel Murguía Romero
7. Datos del trabajo escrito
    - Desarrollo de una extensión a GeoNode bajo el framework Django para implementar un atlas digital en línea.
    - 67 p.
    - 2016

Índice general

1. Introducción **1**
1.1. Planteamiento del problema 1
1.2. Propósito y objetivos del proyecto 5
1.3. Organización 6

2. Marco Teórico **7**
2.1. Definiciones y conceptos básicos en el ámbito de la cibercartografía . . . 7
2.2. Servidores de mapas en línea 11
2.2.1. Infraestructura de datos espaciales (IDE) 11
2.3. Importancia de los estándares de interoperabilidad espacial 12
2.4. Servicios web OGC (Open Geospatial Consortium) 12
2.4.1. Estándares OGC 12
2.4.2. Servicio web 13
2.4.3. Web Map Service (WMS) 13
2.4.4. Web Feature Service (WFS) 13
2.4.5. Catalog Service for the Web (CSW) 14
2.4.6. Web Map Tile Service (WMTS) 14
2.4.7. Web Coverage Service (WCS) 15
2.4.8. Web Processing Service (WPS) 15
2.5. Clientes web geoespaciales 16
2.5.1. Aplicación de mapeo web enriquecida 16
2.5.2. OpenLayers 16
2.6. OpenStreetMap 17
2.7. MapQuest 18
2.8. Catálogos web geoespaciales 19
2.8.1. Pycsw 19
2.8.2. Deegree 19
2.9. Plataformas colaborativas geoespaciales 19
2.10. GeoNode 20

3. Análisis y Diseño del Atlas **21**
3.1. Introducción . . . 22
3.1.1. Patrones de diseño . . . 22
3.1.1.1. Patrón Modelo-Vista-Controlador (MVC) . . . 22
3.1.1.2. Patrón Model-View-Template (MVT) . . . 24
3.1.2. Python . . . 24
3.1.3. Framework . . . 24
3.1.3.1. Django . . . 25
3.1.4. GeoNode . . . 26
3.1.5. Apps de GeoNode . . . 27
3.1.6. Arquitectura . . . 28
3.1.7. GeoServer . . . 30
3.1.8. GeoExplorer . . . 31
3.1.9. Apache . . . 31
3.1.10. PostgreSQL . . . 32
3.1.11. Bibliotecas geoespaciales de Python . . . 32
3.1.12. jQuery . . . 33
3.1.13. Bootstrap . . . 33
3.1.14. Ext JS . . . 33
3.1.15. GeoExt . . . 34
3.2. Análisis de Requerimientos . . . 35
3.2.1. Objetivo . . . 35
3.2.2. Requerimientos funcionales . . . 35
3.2.3. Requerimientos no funcionales . . . 36
3.3. Diseño e implementación del atlas . . . 37
3.3.1. Prototipo inicial . . . 37
3.3.2. Arquitectura del sistema . . . 39
3.3.3. Diseño del comportamiento . . . 41
3.3.4. Diseño de la interfaz . . . 42
3.3.5. Diseño del esquema de datos . . . 45
3.3.6. Implementación . . . 47
3.3.7. Integración con GeoNode . . . 47
3.3.8. Ambiente de desarrollo . . . 48
3.3.9. Carga del atlas . . . 48
3.3.10. Vista final . . . 49

4. Resultados y Conclusiones **55**
4.1. Introducción . . . 55
4.1.1. Problemas encontrados . . . 56
4.1.2. Resultados . . . 57
4.1.3. Conclusiones . . . 57
4.1.4. Trabajo a futuro . . . 58

Anexos **60**

A. Anexo I: Instalación de GeoNode para el desarrollo **61**

CAPÍTULO 1

Introducción

1.1. Planteamiento del problema

El volcán Chichón se localiza en el estado de Chiapas, una región del sudeste de México que se encuentra habitada principalmente por indígenas de la etnia Zoque. El Chichón es el más joven de los volcanes que forman el Arco Volcánico chiapaneco. Antes de la erupción de 1982, no había información histórica sobre su actividad; sin embargo, algunos lugareños refieren que hubo una erupción hace aproximadamente 100 años [1].

La madrugada del 28 de marzo de 1982, el volcán Chichonal o Chichón, empezó su actividad volcánica la cual se extendió hasta el 4 de abril y dejó como saldo un pueblo sepultado, más de 2000 muertos, 20 mil desplazados y miles de hectáreas de cultivo dañadas[2].

Figura 1.1: Muros de la iglesia de Francisco León Foto: Robert Tilling, 1982

Fuente: https://geolocation.ws/v/P/69675782/muros-de-la-iglesia-de-francisco-len-by/en

Después del 3 de abril de 1982, sucedió el episodio más violento. El volcán arrojó oleadas y flujos piroclásticos que formaron una columna de cenizas de hasta 24 kilómetros de altura. La nube que se formó se desplazó hacía el oeste[3]. Estos hechos devastaron un área de aproximadamente 10 km alrededor del mismo y cubrió el sudeste de México con ceniza. Esto resultó ser el peor desastre volcánico registrado en la historia de México [1].

Figura 1.2: Fotografía tomada antes de la erupción del pueblo de Francisco León. En un circulo rojo se encuentra señalada la iglesia. Foto: Ricardo Meléndez Urista

Fuente: http://www.jornada.unam.mx/2002/01/28/cien-tzotzil.html

[1]Martin del Pozo, *op. cit.*, pags 21-31.

Figura 1.3: Imagen tomada después de la erupción. En el circulo rojo se señala donde se ubicaba la iglesia. Foto: W. A. Duffield

Fuente: http://www.jornada.unam.mx/2002/01/28/cien-tzotzil.html

La erupción del volcán Chichón en el año de 1982 provocó la dispersión de las comunidades, la migración a diversas partes de Chiapas y de la república. En las zonas de mayor impacto de la actividad volcánica causó la desaparición de la flora y fauna así como también de los cultivos de maíz, cacao, frijol y café, además de otras afectaciones en el suelo como deslizamientos de tierra y la obstrucción en ríos [2].

Figura 1.4: Pueblo de Francisco León, después de la erupción del Volcán Chichonal Foto R. I. Tilling 1982.

Fuente: http://www.panoramio.com/photo/68974774

Tiempo después, la zona se ha ido recuperando y en ella crecen helechos, orquídeas, líquenes, especies arbustivas y pinos. Además, el efecto de las cenizas como fertilizante ha ayudado a recuperar el suelo, por lo que los pobladores ven los resultados de la recuperación y con esto la posibilidad de regresar a sus tierras. De igual, forma estas zonas se vuelven visitadas por científicos, turistas de aventura, estudiantes, la población en general y la población local. Lo anterior trae como consecuencia la reactivación gradual de las actividades económicas en la zona[3].

[2] Alcántara, *et. al.*, 2013 p. 6
[3] Ibídem

Las zonas cercanas al volcán Chichón son de importancia, ya que al igual que otras regiones de Chiapas, se encuentran expuestas a fenómenos naturales, causas socioculturales, sin olvidar las erupciones volcánicas y la actividad sísmica. Las consecuencias de estos fenómenos afectan en especial a las comunidades socialmente vulnerables de la población chiapaneca[4].

Debido a las consecuencias que tiene el vulcanismo sobre las civilizaciones, el entendimiento de la actividad volcánica y sus repercusiones se vuelven un tema obligado en las regiones donde se encuentran volcanes activos[5].

Figura 1.5: La devastación por la erupción del volcán Chichonal en 1982 Foto: René Canul

Fuente: http://www.proceso.com.mx/?p=368333

Por las razones anteriores se realizó el libro *La región del volcán Chichón, Chiapas: un espacio potencial para su protección, conservación y desarrollo sustentable*, el cual reúne información importante sobre el volcán Chichón: desde aspectos ambientales, de recursos naturales, los rasgos geológicos y de aspectos sociales y económicos de la región; los cuales conviven dentro de un espacio que se considera debe ser protegido, conservado y preparado para su potencial desarrollo sustentable. Se trata de un esfuerzo científico y editorial, producto de acciones de investigación e intercambio académico desarrollado en el marco de un convenio de colaboración entre el Instituto de Geografía de la Universidad Nacional Autónoma de México y la Universidad de Ciencias y Artes del Estado de Chiapas [6].

El atlas impreso es resultado de una colaboración entre la Universidad Nacional Autónoma de México mediante sus institutos de Geografía y Geofísica y de la Universidad de Ciencias y Artes del Estado de Chiapas, a través del Centro de Investigación en

[4] *Ibídem*
[5] *Ibídem*
[6] *Ibídem*

Gestión de Riesgos y Cambio Climático. La obra impresa se encuentra dividida en cinco partes[7] :

- I. El Espacio natural,
- II. Indicadores climáticos,
- III. Geomorfometría,
- IV. Amenazas de origen natural y socio-natural,
- V. Caracterización de la vulnerabilidad socio-económica.

La obra reúne información importante que podrá ser usada por las instituciones oficiales como protección civil para la prevención de riesgos. También sería de utilidad para las universidades y sus programas de investigación[8].

1.2. Propósito y objetivos del proyecto

Una vez finalizado el atlas y dado que se contaba con los insumos digitales surgió la idea de extenderlo haciendo una versión digital en línea con dos requerimientos centrales: 1) respetar la riqueza de información y calidad cartográfica de la versión impresa y 2) extender la audiencia del atlas ofreciendo los recursos a través de servicios Web interoperables y agregando elementos de visualización interactiva[4].

La información completa estará disponible a través de un sitio web de mapas interactivos. Adicionalmente y para dar sustento a la idea de un atlas digital, se ofrecerán funciones para agregar, borrar y extender los contenidos basada en roles y permisos. Haciendo uso de servicios web, se ofrecerá la información a dispositivos de escritorio y móviles[9].

Para lograr lo anterior, se pretende adecuar la funcionalidad del proyecto de *software* libre *GeoNode*, mediante una *app* que permita la administración de un nivel adicional de organización, el de "atlas"que agrupará un conjunto de temas, que a su vez se organizan en "temas específicos".[10].

Se espera que la solución propuesta contribuya a la construcción colaborativa de una Infraestructura de Datos Espaciales[5]. Esta se basará en la interoperabilidad de acervos y servicios entre varias instituciones y organizaciones con interés en la zona mediante servicios web interoperables, además que facilite la integración y estandarización de datos, proporcionando un proceso claro y sencillo para la carga de información geográfica, la compilación de metadatos, la difusión de documentos y la creación de mapas[11].

[7] *Ibídem*
[8] *Ibídem*
[9] *Ibídem*
[10] *Ibídem*
[11] *Ibídem*

1.3. Organización

La estructura de este reporte se divide 3 secciones. En la primera se hace una introducción a los conceptos y herramientas de software en el ámbito de la cibercartografía. La segunda parte está dividida en tres subsecciones. En la primer subsección se dan conceptos de desarrollo de *software* junto con otras herramientas utilizadas para el atlas. En la segunda se mencionan las características que debe tener el atlas y la última de estas se habla sobre el diseño y la implementación. En la parte final se mencionan el estado actual del proyecto, los problemas encontrados y las conclusiones.

CAPÍTULO 2

Marco Teórico

2.1. Definiciones y conceptos básicos en el ámbito de la cibercartografía

Cibercartografía es el análisis, presentación y comunicación de la información geográfica en una amplia variedad de formas de interés y de uso para la sociedad que pueden ser formatos interactivos , dinámicos, multimedia, multisensorial o multidisciplinarios[6].

Los **datos geoespaciales**, también llamados geodatos o datos georeferenciados, describen la ubicación de un lugar en la Tierra, mediante sus atributos brindan una imagen completa del mundo físico, tanto en términos de espacio como de tiempo. Estos pueden almacenarse en distintos tipos de formatos para después desplegarse como imágenes.

Rasgo geográfico, es una representación de un objeto del mundo real puesto en un mapa[7].

Capa es la representación visual de un conjunto de datos que se despliegan sobre cualquier mapa digital. Conceptualmente una capa es un corte o estrato de un área geográfica en particular como se puede apreciar en la Figura 3.4, cada una equivale a uno de los elementos en la leyenda de un mapa impreso[8].

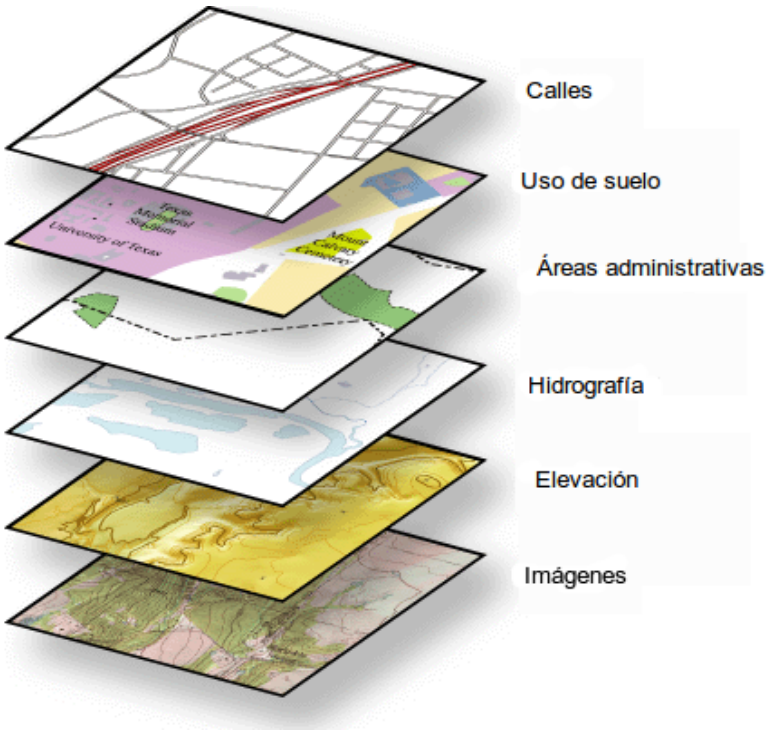


Figura 2.1: Ejemplo de capas

Fuente: http://resources.arcgis.com/es/help/getting-started/articles/026n0000000q000000.htm

Las capas se agregan en mapas. Un **mapa**, que en el área de la cartografía es cualquier representación gráfica de información espacial o geográfica de las relaciones espaciales de entidades dentro de un área[9]. Un mapa impreso también incluye elementos adicionales, dispuestos y organizados en una página[10]. En la Figura 2.2 se ilustran estos elementos y su distribución sobre el mapa.

Los componentes son:

- Leyenda : contiene el significado de los símbolos en el mapa;
- Marco del mapa: es el mapa en cuestión;
- Fuente de datos: indica de donde provienen los datos;
- Retícula: es una red de lineas de longitud y latitud sobre un mapa que relaciona puntos en un mapa con sus verdaderos lugares en la tierra;
- Escala : esta parte que dice como interpretar las distancias sobre el mapa;
- Proyección: indica el sistema de representación gráfico que se uso en el mapa;
- Flecha de Norte : es la parte del mapa que muestra en el mapa donde se encuentra el norte, sur, este y oeste;
- Mapa de localización: es un mapa que sirve para mostrar donde se encuentra la zona del mapa;

- Titulo : esta parte del mapa indica sobre que trata el mapa.

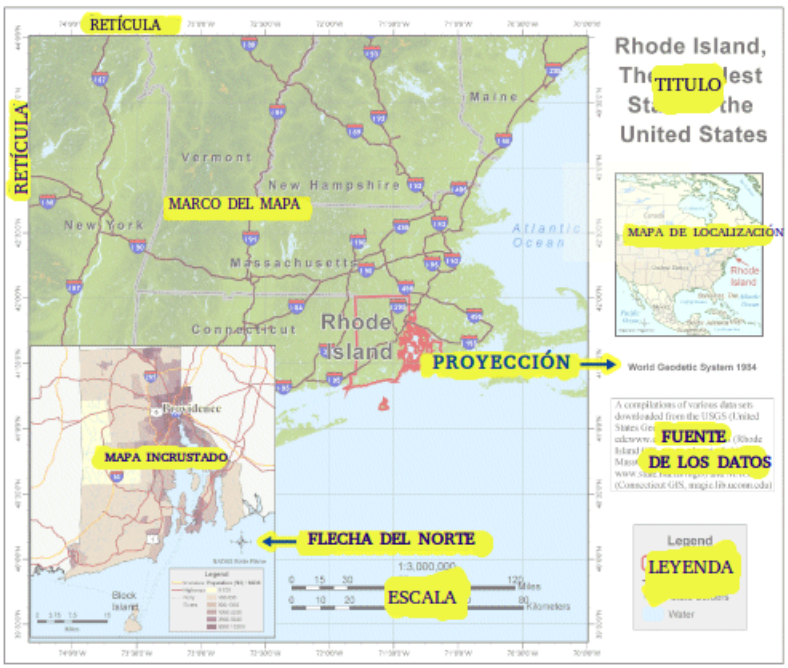


Figura 2.2: Elementos del mapa

Fuente: http://resources.arcgis.com/es/help/getting-started/articles/026n0000000q000000.htm

Shapefile, es un formato que sirve para almacenar la ubicación geométrica y la información de atributos de las entidades geográficas. Actualmente su uso es un estándar para el intercambio de información espacial entre todo tipo de sistemas, y es el formato en el que la mayoría de los datos geoespaciales comerciales se distribuyen Fue desarrollado para usarse en el paquete de *software* ARC/INFO GIS(*Geographic Information System*) de ESRI[11].

Un *shapefile* se encuentra compuesto por al menos tres archivos. Cada uno de estos comparte el mismo nombre. A continuación se listan los archivos necesarios dentro de un archivo *shape*[12]:

- .shp: es el archivo principal y guarda la geometría de la entidad; es necesario.
- .shx: en este archivo se almacena el índice de la geometría de la entidad; es necesario.
- .dbf: es el archivo que almacena la información sobre atributos de las entidades; es necesario.

Los siguientes archivos son opcionales:

- .sbn y .sbx: son los archivos que almacenan el índice espacial de las entidades; es opcional

- prj: es el archivo que almacena la información sobre el sistema de coordenadas; se utiliza en ArcGIS y es opcional.
- xml: es el archivo de metadatos de ArcGIS y almacena información sobre el *shapefile* es opcional
- .cpg: es un archivo que se puede utilizar para especificar la página de código que identifica el conjunto de caracteres que se va a utilizar; es opcional

Los datos tipo ***raster***, es cualquier tipo de imagen digital que en su forma mas simple, consisten de una matriz de píxeles que se encuentra organizada en renglones y columnas donde cada píxel o celda contiene un valor de un color representando información. O este valor puede ser un valor discreto o un valor continuo. Éste tipo de imagen se usa en las fotografías digitales aéreas, imágenes satelitales y digitales, incluso en mapas escaneados. También se utiliza para representar información concerniente a la altura o almacenar información referente al reflejo de una longitud de onda en particular de una zona[13]. En la Figura 2.3 se ejemplifica una imagen *raster* y la malla de píxeles que la componen.

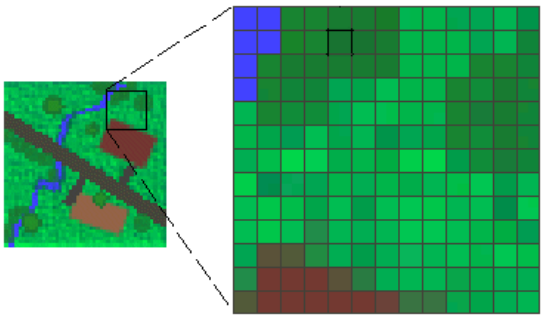


Figura 2.3: Ejemplo de una imagen *raster*

Fuente: http://webhelp.esri.com/arcgisexplorer/2500/es/

El modelo **vectorial** es una imagen digital formada por primitivas gráficas(puntos, lineas y polígonos) que puede ser utilizada para representar objetos geográficos(ríos, caminos, vegetación, etc.)[14]. La Figura 2.4 ilustra como se usan estas primitivas para representar objetos en una imagen.

Cubierta (*coverage*), es información geoespacial que representa algún fenomeno espacio/tiempo [15].

Un **atlas** es una colección de mapas o conjuntos de datos en forma de libro que contienen información estructurada de manera jerárquica. Los atlas han emergido en los últimos años como un producto digital que toma ventaja de la accesibilidad y funcionalidad que el ambiente en línea les provee. Entre las principales categorías de atlas están los geográficos, históricos, nacionales o regionales, topográficos, anatómicos y temáticos[16].

Figura 2.4: Ejemplo de un vector

Fuente: http://resources.arcgis.com/es/help/getting-started/articles/026n0000000p000000.htm

2.2. Servidores de mapas en línea

Un servidor de mapas (geospatial server) es un *software* que se encarga de distribuir datos geoespaciales en internet mediante los estándares de la OGC (Open Geoespatial Consortium), los cuales se explicarán más adelante. Estos servidores proveen a clientes web los datos como rasgos o cubiertas dependiendo del estándar que se use [17].

2.2.1. Infraestructura de datos espaciales (IDE)

Una infraestructura de datos espaciales (en Inglés *Spatial Data Infrastructure*) se define por lo general como un conjunto de tecnologías, políticas, estándares y acuerdos institucionales orientadas a facilitar, almacenar, procesar y mejorar el acceso distribuido e interoperable a colecciones de datos geoespaciales [18].

Una infraestructura de datos espaciales permite optimizar el acceso a la información geográfica. Esto se logra mediante acciones coordinadas de naciones y organizaciones que promueven el conocimiento y la implementación de políticas complementarias, los estándares en común y mecanismos efectivos para el desarrollo de la disponibilidad e interoperabilidad digital geográfica para apoyo en la toma de decisiones en todas las escalas para múltiples propósitos. Estos actos abarcan las políticas, ámbitos organizacionales, datos, tecnologías, estándares, mecanismos de distribución, recursos humanos y financieros necesarios para garantizar que éstos no se vean obstaculizados en lograr sus objetivos a un nivel nacional y regional [5].

2.3. Importancia de los estándares de interoperabilidad espacial

Vivimos en un mundo que cambia constantemente gracias a la comunicación, esto significa que cada vez esta toma más importancia en nuestra vida cotidiana. La revolución de la comunicación llegó con el internet, que debe en gran medida su éxito, a los estándares abiertos de interoperabilidad. A la capacidad de un sistema o componente de interactuar con otro a mediante interfaces estandarizadas sencillas y bien documentadas se le llama **interoperabiliad**. Los estándares de la OGC (*Open Geoespatial Consortium*), se han vuelto parte de las buenas prácticas de los SDI alrededor del mundo. Los estándares bajan costos y disminuyen el tiempo requerido para la integración de sistemas. Con el amplio uso de los estándares basados en servicios web se logra tener un bajo acoplamiento entre sistemas, lo que ahorra la necesidad de conocer los detalles de implementación de los componentes con los que se desea interactuar[1].

2.4. Servicios web OGC (Open Geospatial Consortium)

El Open Geoespatial Consortium es una organización internacional sin fines de lucro fundada en 1994. En en esta organización participan universidades, agencias de gobierno y empresas desarrolladoras de *software* geoespacial. La OGC se enfoca en desarrollar y promover estándares abiertos para servicios de geolocalización, mecanismos para la interoperabilidad de datos geoespaciales y sensores[19].

Está organización produce principalmente estándares o protocolos de comunicación, estos son especificados en documentos que contienen interfaces y las reglas para su codificación. Para la elaboración de las especificaciones se lleva a cabo un proceso en el que los protocolos son consensuados, discutidos, aprobados y puestos a prueba por sus miembros. Estos se diseñan para que los datos y servicios espaciales sean accesibles y útiles para cualquier tipo de aplicación [20].

2.4.1. Estándares OGC

Los estándares OGC son documentos técnicos que detallan interfaces o codificaciones. Los desarrolladores de *software* usan estos documentos para construir interfaces abiertas y escribir código interoperable para sus productos y servicios. Estos son la principal producción de la OGC y son desarrollados por los miembros para resolver los retos específicos que plantea la interoperabilidad. Una lista completa y actualizada con los más de 30 estándares OGC se encuentra disponible en la página de la OGC. Todos estos están al alcance de cualquier persona, sin costo y con su respectiva documentación[21]. Para este reporte solo se explican las características de los estándares que implementa GeoServer, del cual hablaremos más adelante.

[1]Luis Bermudez, *op. cit.*, p 107

2.4.2. Servicio web

Un **servicio web** es una tecnología que permite que distintos dispositivos o aplicaciones se comuniquen por el internet en una forma que no depende de la plataforma ni del lenguaje de programación. Un servicio web esta compuesto por una interfaz de software que describe un conjunto de operaciones a las cuales se puede acceder por la red usando XML(*eXtensible Markup Language*). Para comunicarse usa protocolos basados en XML con el objetivo de describir operaciones a ejecutar o datos a intercambiar con otro servicio web [22].

2.4.3. Web Map Service (WMS)

El servicio WMS es un protocolo de la OGC que produce imágenes de mapas referenciados espacialmente a partir de información geográfica. Este estándar especifica operaciones para obtener una descripción de los mapas ofrecidos por un servidor y consultar sobre los rasgos desplegados en un mapa. El protocolo es aplicable para obtener representaciones pictóricas de los mapas en un formato gráfico, no aplica para la obtención de los datos de los rasgos o coberturas[23].

Este servicio cuenta con las siguientes operaciones[2]:

- *GetCapabilities*(obligatorio): devuelve los metadatos del nivel de servicio;
- *GetMap* (obligatorio): devuelve la imagen de un mapa con parámetros bien definidos;
- *GetFeatureInfo*(opcional): devuelve información sobre las características de los rasgos en un mapa.

2.4.4. Web Feature Service (WFS)

El servicio *Web Feature Service* (WFS) es un estándar OGC de entrega de datos que define operaciones para acceder a los datos y manipularlos. La interfaz de WFS permite consultar, editar y descargar datos geoespaciales. Los datos de rasgos geoespaciales deben ser codificados en GML (Geography Markup Language) [24], la cual es una especificación OGC basada en XML que permite el almacenamiento, transporte, procesamiento y transformación de datos geoespaciales[25]. La OGC define las siguientes operaciones para el estándar WFS[3]:

- *GetCapabilities*(obligatorio): devuelve metadatos de nivel de servicio;
- *DescribeFeatureType*(obligatorio): devuelve la descripción de los tipos de rasgos;

[2]Número de referencia del documento OGC 06-042, Versión: 1.3.0
[3]Número de referencia del documento OGC 04-094, Versión: 1.1.0

- *GetFeature*(obligatorio): devuelve los rasgos pedidos;
- *Transaction*(opcional): permite editar rasgos (crear, actualizar y borrar);
- *LoockFeature*(opcional):previene la edición de rasgos mediante un bloqueo a largo plazo.

2.4.5. Catalog Service for the Web (CSW)

Catalog Service for the Web (CSW), es un servicio web que se usa para exponer catálogos de registros geoespaciales en internet. CSW define interfaces entre clientes y servicios para descubrir y obtener datos geoespaciales e información sobre servicios de metadatos con HTTP. Con los servicios de catálogo se puede publicar y buscar información de recursos geoespaciales y servicios relacionados[15]. Este servicio cuenta con las siguientes operaciones[4]:

- *GetCapabilities*(obligatorio): devuelve los metadatos de nivel de servicio;
- *DescribeRecord*(obligatorio): devuelve información sobre el modelo de registros;
- *GetDomain*(opcional): devuelve el rango de valores de un registro;
- *GetRecords*(obligatorio): devuelve registros y devuelve sus identificadores;
- *GetRecordById*(obligatorio): devuelve un registro de acuerdo a su identificador;
- *Transaction*(opcional): crea, edita y borra registros de metadatos;
- *Harvest*(opcional): crea o actualiza registros de metadatos.

2.4.6. Web Map Tile Service (WMTS)

El estándar *Web Map Tile Service* (WMTS) complementa al estándar *Web Map Service* de la OGC. El propósito del estándar WMTS es habilitar un servicio que se oriente al desempeño y a la escalabilidad. Por lo tanto los servidores que implementan WMTS pueden devolver imágenes de mapas divididas en teselas (tiles). Una **tesela** es una representación gráfica rectangular de datos geoespaciales que a menudo también son parte de otro conjunto de teselas que cubren una extensión contigua y comparten información. Se colocan contiguamente para formar un mapa. de forma rápida. Una forma de lograr esto es usar piezas o teselas ya procesadas, las cuales se almacenan en el servidor y opcionalmente pueden almacenarse en el cache local y no requieren ninguna manipulación o geoprocesamiento[26].

La interfaz WMTS permite al cliente obtener tres tipos de datos, ya sea en respuesta a la petición de un recurso o a un procedimiento. Los operaciones que define la interfaz son las siguientes[5]:

[4]Número de referencia del documento OGC 07-110r4, Versión: 1.0.1
[5]Número de referencia del documento OGC 07-057r7, Versión: 1.0.0

- *GetCapabilities*(obligatoria): permite al cliente obtener documentos de metadatos de un servidor;
- *GetTile*(obligatoria): permite al cliente obtener una tile de una matriz en particular en un formato predefinido;
- *GetFeatureInfo*(opcional): permite obtener información sobre rasgos presentes de píxel en particular de un tile en un mapa.

2.4.7. Web Coverage Service (WCS)

El estándar *Web Coverage Service* (WCS) admite la obtención electrónica de información geográfica geoespacial como datos tipo *raster* representando fenómenos espacio/-tiempo que son accesados de tal forma que el procesamiento se hace del lado del cliente. El servicio WCS permite al cliente descubrir y examinar datos con restricciones espaciales u otros criterios de consulta, también provee los datos reales junto con su descripción detallada y semántica original, que no solamente puede ser representada en un mapa sino también interpretada y extrapolada[27]. El estándar posee las siguientes operaciones[6]:

- *GetCapabilities*(obligatoria): permite al cliente obtener información acerca del servicio. La respuesta contiene la descripción en la que avisa de las características operacionales y no operacionales del servicio;
- *DescribeCoverage*(obligatoria): provee una lista de identificadores de cubiertas, para cada identificador corresponde una descripción;
- *GetCoverage* (obligatoria): Permite solicitar una cobertura mediante un cuadro delimitador y regresa las coberturas que se encuentren dentro del dominio de la envolvente.

2.4.8. Web Processing Service (WPS)

El estándar *Processing Service* (WPS) provee al cliente acceso a operaciones preprogramadas o modelos calculados que operan sobre recursos referenciados espacialmente. Estos pueden ser requeridos y obtenidos por el servicio, usar formatos de imagen o formatos de intercambio como *Geography Markup Language* (GML). Los cálculos pueden ser simples como sustraer un conjunto de números referenciados espacialmente de otros. Cabe señalar que lo importante de estandarizar estos procesos es reducir la cantidad de programación requerida y facilitar la implementación y adopción de nuevos servicios. El estándar WPS especifica tres operaciones que un cliente puede usar hacia un servidor que implemente el estándar[28]. Las operaciones son[7]:

[6]Número de referencia del documento OGC 09-110r4, Versión: 2.0.1
[7]Número de referencia del documento OGC 05-007r7, Versión: 1.0.0

- *GetCapabilities*(obligatoria): esta operación permite al cliente hacer una petición y recibir servicio de metadatos, documentos que describen las habilidades de un servidor especifico. Con la operación GetCapabilities se obtienen los nombres y la descripción general de cada proceso ofrecido por por el servidor que implementa WPS;

- *DescribeProcess*(obligatoria): esta operación permite al cliente hacer peticiones y recibir información detallada acerca de los procesos que pueden correr en la instancia del servicio, incluyendo las entradas requeridas, sus formatos permitidos y las salidas que se producen;

- *Execute*(obligatorio): esta operación permite al cliente correr un proceso especificado por el estándar WPS, usando valores de los parámetros de entrada proporcionados y devolviendo los resultados producidos.

2.5. Clientes web geoespaciales

Los **clientes web geoespaciales** son visualizadores que integran en una sola interfaz, varias conexiones simultaneas a distintos servidores, estos visualizadores se encargan de proveer acceso a los datos geoespaciales, permiten buscar mapas, hacer zoom en un mapa, interactuar, hacer consultas, apagar o encender capas *GeoWeb* que usen estándares de la OGC [8].

2.5.1. Aplicación de mapeo web enriquecida

Una aplicación de mapeo web enriquecida es una aplicación web para hacer mapas que tiene la mayoría de las características de las aplicaciones de escritorio tradicionales. Estas aplicaciones se ejecutan en un navegador web y por medio de complementos, bibliotecas como JavaScript o máquinas virtuales se agregan características adicionales [29].

2.5.2. OpenLayers

Es una biblioteca de JavaScript que provee funciones para insertar mapas interactivos en un páginas web. El desarrollo de OpenLayers inició en el año de 2005 por la compañía estadounidense MetaCarta [9] con el propósito de construir el equivalente a Google Maps en código abierto [10].

OpenLayers implementa estándares OGC para métodos geográficos de acceso a los datos como WMS, WFS, incluyendo WFS-T (*Web Feature Service Transactional*) y WCS,

[8]Marco Minghini, *op. cit.*, p. 71
[9]http://www.metacarta.com
[10]Marco Minghini, *op. cit.*, p. 72

también soporta formatos de datos tales como GeoRSS(*Geographic Really Simple Syndication*), KML(*Keyhole Markup Language*) y GeoJSON(*Geographic JavaScript Object Notation*). La filosofía de OpenLayers es separar las herramientas de mapas de los datos de los mismos, así todos los instrumentos pueden operar sobre todas las fuentes. Además incluye integración con *OpenStret Map*, *Google Maps*, *Yahoo! Maps* y *Bing Maps* cuya disponibilidad permite crear composición de mapas [11]. La versión actual de OpenLayers es la 3.6[30]. En la Figura 2.5 se muestra la apariencia de un mapa incrustado en una página web usando OpenLayers. Para este proyecto se usan mapas interactivos que son hacen uso de OpenLayers.

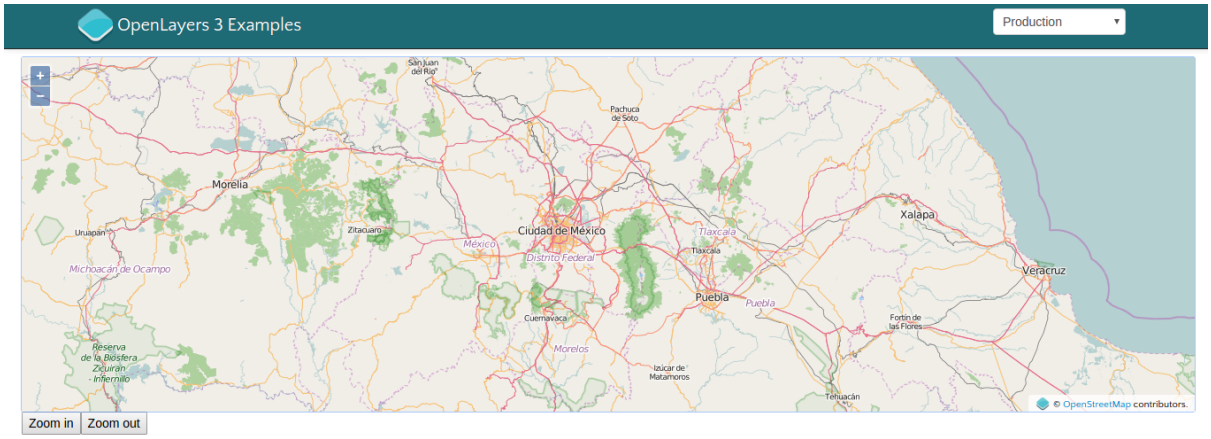


Figura 2.5: Ejemplo de un mapa con OpenLayers

Fuente: http://openlayers.org/en/v3.2.1/examples/overviewmap-custom.html

2.6. OpenStreetMap

OpenstreetMap(OSM), es un proyecto colaborativo para crear y proveer acceso a mapas libres y editables[31]. Suministra datos geoespaciales a sitios web, aplicaciones móviles y dispositivos de hardware. OpenStreetMap es apoyado por una gran comunidad de usuarios que colaboran al añadir mapas y datos sobre caminos, senderos, cafeterías, estaciones de tren, baños y muchos lugares al alrededor del mundo como se puede ver en la Figura 2.6[12].

Las ventajas de OpenStretMap son las siguientes[13]:

- Valora el conocimiento local de los colaboradores. Estos utilizan imágenes aéreas, dispositivos GPS, mapas y otras fuentes de datos libres para verificar que los datos de OSM sean precisos y estén actualizados;
- Es un proyecto impulsado por los usuarios. Entre los cuales se encuentran profesionales de los sistemas de información geográfica, ingenieros que mantienen los

[11] *Ibídem*
[12] *Ibídem*
[13] *Ibídem*

servidores del proyecto, gente que mapea zonas afectadas por desastres naturales y mucho más,

- Es un acervo libre. Los datos en OSM pueden ser usados para cualquier propósito, siempre y cuando se le dé el respectivo crédito a OSM.

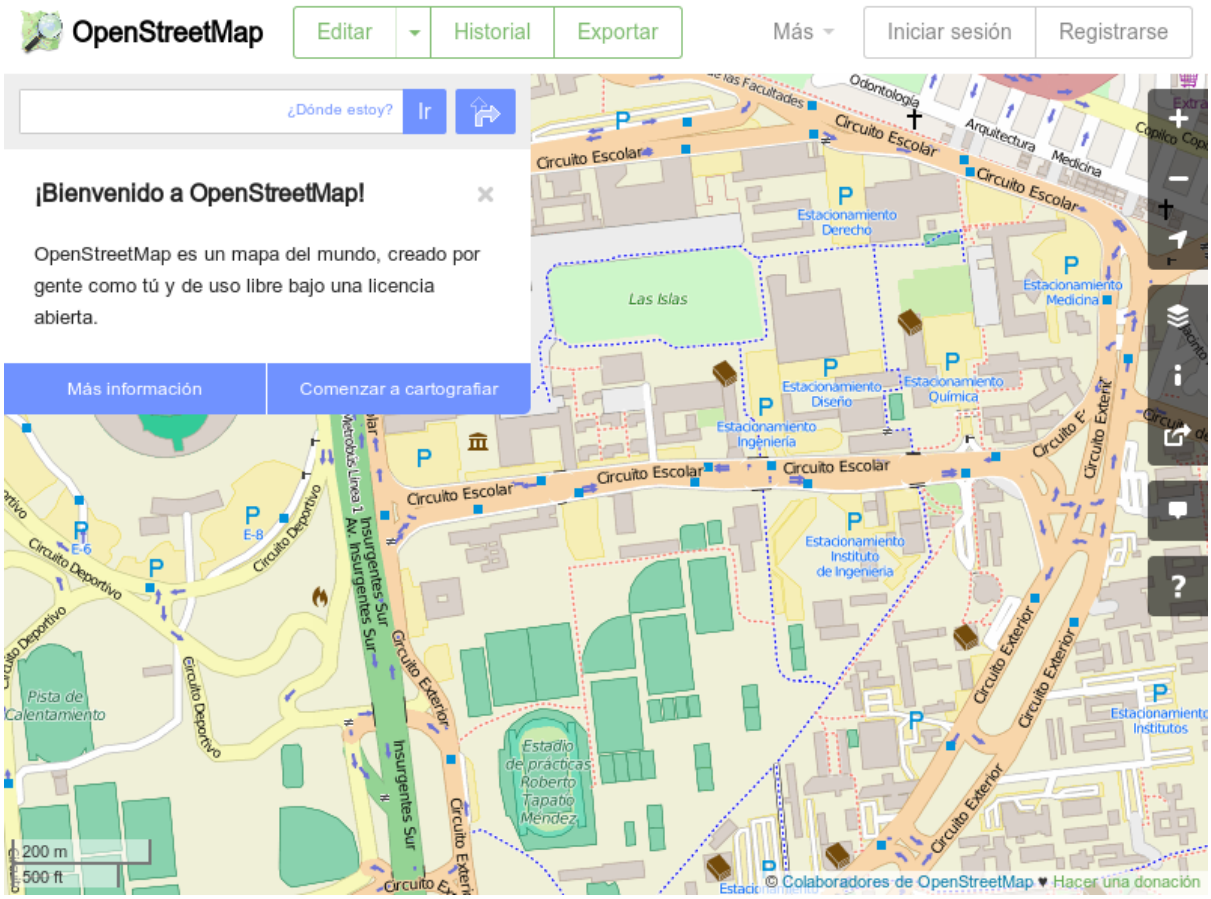


Figura 2.6: Ejemplo de un mapa usando OpenStreetMap

Fuente: https://www.openstreetmap.org/#map16/19.3271/99.1870

2.7. MapQuest

MapQuest es una empresa propiedad de *AOL* que brinda servicio de mapeo en línea gratuito. Su objetivo es proveer mapas y direcciones certeras a millones de personas cada día, facilitar a sus usuarios encontrar información local, descubrir lugares nuevos, planear rutas, personalizarlas y compartirlas con otras personas[32].

2.8. Catálogos web geoespaciales

Un catálogo web geoespaciales es *software* que se ejecuta en un servidor que provee operaciones básicas para buscar, editar y publicar metadatos relacionados a información, geoespacial. Algunos catálogos implementan el estándar CSW de la OGC [14].

2.8.1. Pycsw

Pycsw es un servidor que implementa el servicio CSW, escrito en el lenguaje de programación Python (que se explicará más adelante), se ha desarrollado desde el 2010. El servidor Pycsw permite publicar y encontrar metadatos geoespaciales. Proporciona un catálogo de metadatos y componentes basados en estándares de infraestructuras de datos geoespaciales. Pycsw es desarrollado por usuarios y programadores; también es un proyecto OsGeo en incubación. Éste corre para la mayoría de los sistemas operativos más importantes e implementa los estándares WMS, WFS, WCS y todas las operaciones de CSW. Es usado por gobiernos, academias e industrias como un componente incrustado en portales geoespaciales [33].

2.8.2. Deegree

Deegree es un proyecto de *software* libre basado en Java para Infraestructuras de Datos Espaciales y para la web geoespacial. Implementa el estándar, ISO/TC 211 de la OGC. Se diseñó para ser una herramienta interoperable. Es un catálogo que almacena metadatos conforme al estándar ISO 19115 diseñado para describir información y servicios geográficos y también conforme al estándar ISO 19119 que sirve definir e identificar los patrones de arquitectura para interfaces de servicios en la información geográfica. *Deegree* incluye componentes para la gestión de datos geoespaciales, incluyendo acceso a datos, visualización, descubrimiento y seguridad. Los estándares abiertos están en el núcleo de *Deegree* [34].

2.9. Plataformas colaborativas geoespaciales

Las plataformas colaborativas geoespaciales son sistemas web 2.0, estas son aquellas aplicaciones que aprovechan las ventajas del internet como la colaboración, el compartir la información y la interoperabilidad. También entregan actualizaciones constantemente, servicios que ayudan a que más gente los utilice, consumiendo y reusando información de múltiples fuentes y usuarios [35]. Las plataformas colaborativas geoespaciales complementan a las infraestructuras de datos geoespaciales con componentes sociales que promueven la colaboración, publicación y creación de mapas. Estos integran en un solo paquete coherente diferentes servicios geoespaciales tales como: servidores de mapas, clientes ligeros,

[14] Marco Minghini, *op. cit.*, p. 74

servicio de catálogo y herramientas de colaboración. Constituyen una forma de fusionar las ventajas de las IDEs con las capacidades de generación colaborativa de contenidos de la Web 2.0 [15].

2.10. GeoNode

GeoNode[16] es una plataforma web que permite administrar, crear y publicar información geoespacial. Esta conformado por proyectos *open-source* maduros, y estables bajo una interfaz consistente y sencilla que permite a los usuarios (con un poco de práctica) compartir datos y crear mapas interactivos[17].

GeoNode habilita la administración sencilla y eficiente de permisos a los diferentes recursos. También cuenta con perfiles de usuario, comentarios y un sistema de rating. Él sistema provee una herramienta para el desarrollo de información. Esta diseñado como una plataforma que es posible adecuar y extender para atender necesidades específicas.

Según Eddie Pickle:

> GeoNode es un proyecto de código abierto, inicialmente apoyado por el Banco Mundial (World Bank), OpenGeo y otros programas piloto en El Salvador y Guatemala. A través de este proyecto se busca dar un giro a la idea de infraestructura de datos espacial (SDI) con especial atención en el uso y reuso de datos, alentando la colaboración y promoviendo el intercambio de datos abiertos por defecto. Inicialmente el desarrollo de GeoNode se ha enfocado en servir y visualizar los datos creados por CAPRA (*Central America Probabilistic Risk Assessment*) para evaluar y mitigar el riesgo provocado por desastres naturales.
>
> En su núcleo, GeoNode se encuentra basado en componentes de código abierto como GeoServer, GeoNetwork, Django y GeoExt que proveen una plataforma para que un navegador permita la visualización y el análisis. En lo alto de esta pila, el proyecto ha construido un sistema para componer y visualizar mapas, herramientas para el análisis y documentación. GeoNode también permite la edición de estilos de datos, así como de características colaborativas como un promedio de *likes*, comentarios y etiquetado de mapas, datos y estilos. GeoNode está construido bajo cuatro principios clave. colaboración, distribución, cartografía y colección de datos [36].

[15] Marco Minghini, *op. cit.*, p. 82
[16] http://geonode.org
[17] Marco Minghini, *op. cit.*, p. 76

CAPÍTULO 3

Análisis y Diseño del Atlas

En este capítulo se especifican los detalles del diseño del atlas, como lo son el *framework*, los componentes del atlas, el esquema de la base de datos, los prototipos de pantalla, el comportamiento y la comunicación de los componentes. Para explicar lo anterior se utilizan diagramas de clase que muestran a los objetos que componen al sistema.

En términos generales, el atlas del volcán Chichón funcionará usando los componentes de GeoNode como se ilustra en la Figura 3.1. Se debe extender el esquema de objetos para manejar los elementos de un atlas que consta de: temas generales que se desagregan en temas específicos que se componen de 1 a N hojas. Cada tema específico debe tener uno o más mapas interactivos, con su leyenda, texto en el que se desarrolla el tema, imágenes, gráficas y tablas (potencialmente interactivas)[1].

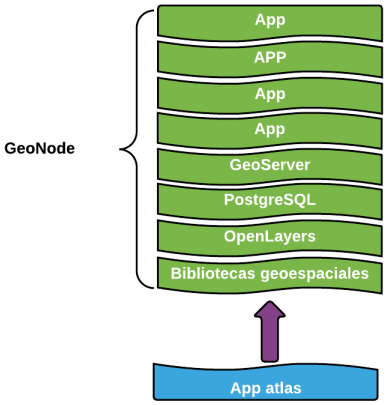


Figura 3.1: La *app* usará los componentes de GeoNode

[1] Javier Osorno, *op. cit.*, p. 1

Se decidió desarrollar el proyecto como una extension de geonode para aprovechar las características que lo hacen una plataforma geoespacial colaborativa. Estas son:

- Permite funciones para la administración de datos y usuarios.
- Habilita la administración de roles y usuarios.
- Permite la visualización y documentación de los mapas.
- Tiene opciones de carga y descarga de datos.
- Permite la interoperabilidad basada en los servicios WMS,WFS,WCS y CSW.
- En su sitio provee documentación para poder extender el proyecto.
- Es un proyecto de código abierto que se encuentra en constante desarrollo.
- Cuenta con foros conformados por una gran comunidad de desarrolladores y usuarios que reportan y corrigen errores en el sistema.
- Es una herramienta colaborativa.
- Al estar escrito con Django es extensible a través de *apps*.

3.1. Introducción

3.1.1. Patrones de diseño

Un patrón representa una solución a un problema recurrente de desarrollo de *software* con un contexto en particular [37]. Además da nombre, abstrae e identifica los aspectos clave de una estructura de diseño común que lo hace útil para crear un diseño orientado a objetos. El patrón identifica a las clases participantes e instancias, sus roles y colaboraciones y la distribución de responsabilidades. Cada patrón se centra en un problema de diseño. Este describe cuando aplicarse, si es posible aplicarse en función de otras restricciones de diseño y sus ventajas y desventajas de de su uso [38].

3.1.1.1. Patrón Modelo-Vista-Controlador (MVC)

El patrón de diseño Modelo-Vista-Controlador describe la arquitectura de un sistema de objetos. Está arquitectura puede ser aplicado a sistemas aislados y también a aplicaciones completas. A diferencia de otros patrones de diseño, el modelo MVC se encuentra menos definido, lo que permite implementaciones alternas. Este es uno de los más importantes modelos de diseño en las Ciencias de la Computación [39].

El patrón Modelo-Vista-Controlador tiene la siguiente estructura[2]:

[2] *Ibídem*

- Objetos del **modelo** de datos que se encargan de encapsular información y proveer mecanismos de persistencia.
- Objetos llamados **controladores**, que se encargan de llevar acabo acciones.
- Objetos llamados **vistas**, las cuales se encargan de observar al esquema de datos, actualizar su apariencia cuando éste cambia, reunir las entradas del usuario para comunicar al controlador que lleve acabo la acción y mostrar información al usuario.

La clave para una implementación exitosa del patrón MVC radica en poner atención en el papel que desempeñan los objetos y la comunicación entre ellos. Las siguientes reglas definen la función de cada objeto[3]:

- Objetos del modelo de datos. Estos se encargan de encapsular y abstraer los datos. Deben estar libres de la lógica e instrucciones de la aplicación.
- Objetos de la vista. Estos objetos se encargan de desplegar la información al usuario.
- Objetos del controlador. Son los que se encargan de ejecutar las acciones en la aplicación.

Otro aspecto importante en el modelo MVC es la comunicación que tienen sus componentes.

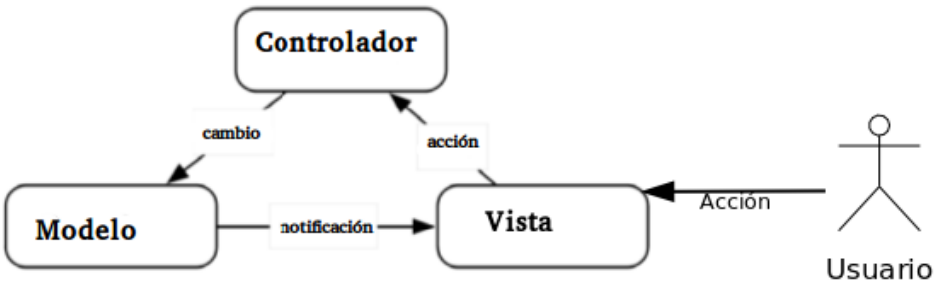


Figura 3.2: Comunicación fundamental en el modelo MVC

Fuente: Learn Objective-Cfor Java Developers

La Figura 3.2 muestra el intercambio de mensajes entre los componentes cuando el usuario provoca un evento, un objeto de la vista se encarga de interpretar la acción y manda un mensaje a un objeto del controlador. A continuación, un objeto del controlador se encarga de llevar a cabo una respuesta, que en la mayoría de las veces involucra mandar mensajes a algún objeto del modelo. Cuando el modelo de datos cambia, se manda un mensaje para que se modifique la vista [4].

[3] *Ibídem*
[4] *Ibídem*

3.1.1.2. Patrón Model-View-Template (MVT)

En el patrón de desarrollo MVT[5]:

- M es la parte del "Modelo", la capa de acceso a los datos. Esta capa contiene cualquier cosa y todo lo relacionado con los datos: como accederlos, validarlos, el comportamiento que deben tener y las relaciones entre datos.
- T es la parte de los *"Templates"*, la capa de presentación. Aquí están las decisiones relacionadas a la representaciones: como algo debe ser desplegado en una página web o en otro tipo de documento.
- V es la parte de la "Vista", la capa de la lógica. Es una función de Python asociada a una URL en particular y define que datos son presentados. Esta capa contiene la lógica que accede al modelo y lo remite a la plantilla apropiada. Se puede pensar que es el puente entre modelos y plantillas.

3.1.2. Python

Python es un lenguaje de programación de alto nivel, multiparadigma (orientado a objetos, funcional o estructural) y multiplataforma creado por el científico de la computación holandés Guido Van Rossum a principios de los años noventa. Es de tipado dinámico y fuerte (las variables pueden cambiar el tipo de contenido, pero el control del contenido es estricto)[40]. Las ventajas de *Python* son[41]:

- Posee una sintaxis limpia y clara (la identación describe el final de los bloques del código);
- Existe una gran cantidad de bibliotecas y *frameworks* para desarrollar con Python(como por ejemplo Django);
- Hay una gran comunidad que se encarga de dar mantenimiento y desarrollar nuevas características.
- Es usado en usado en el computo científico. Cuenta con bibliotecas para las matemáticas, ingeniería y ciencia.

3.1.3. Framework

Un *framework* es una aplicación reusable y semi completa que sirve para producir aplicaciones personalizadas.Los patrones y frameworks se usan en combinación para optimizar el diseño e implementación de aplicaciones, ya que abstraen con éxito estrategias probadas para el desarrollo de software. Estos patrones capturan las características del

[5]Holovaty, *op. cit.*, pags. 73, 74

diseño y de la arquitectura de forma que puede ser comprendida fácilmente por los desarrolladores. Los *frameworks* capturan diseños concretos, algoritmos e implementaciones en algún lenguaje de programación[37].

Los beneficios de usar un *framework* son[6]:

- Reducen los costos de producción y mantenimiento del *software*.
- Permiten la reutilización de diseños y arquitecturas.
- Proveen un estándar de comunicación entre los componentes del *software*.
- Brindan un esqueleto que hereda de los componentes del *framework* y que se puede personalizar.

3.1.3.1. Django

Django es un *framework* web que fomenta el desarrollo rápido, el diseño limpio y pragmático [42]. Django sigue el patrón MVC lo suficientemente cerca, que puede ser llamado un *framework* MVC. Esta es mas o menos la forma en como la M, V y la C cambian en Django:

- M, es la parte de acceso a los datos, es manejada por la capa de la base de datos de Django.
- V, es la parte que escoge que datos y como se se despliegan, es manejado por las vistas y plantillas(*templates*) .
- C, es la parte que asigna una vista en función de la entrada del usuario, es manejada por el propio *framework* mediante la URLconf y mediante una función de python apropiada para la URL dada.

Debido a que la C es manejada por el framework y lo más importante sucede en los modelos, plantillas y vistas, a Django se le ha referido como un framework MVT.

Sus principales características son[43]:

- Un mecanismo llamado ORM (Object Relational Mapper) el cual es un motor de traducción entre Python y código SQL (Structured Query Language). Esta herramienta se encarga de traducir de código en Python que define el esquema de la base de datos así como de la interacción con los datos.
- Sistema Administrador Automático de Contenido (Automatic Content Management System). Con frecuencia es necesario administrar la interfaz con los datos, opciones de seguridad y usuarios así como la restricción de la administración de los datos a un usuario. Django se encarga de construir este sistema de forma automática.

[6] *Ibídem*

- Diseño de URL. *Django* tiene bajo acoplamiento y sigue el modelo MVT esto permite separar la definición de la URL del recurso. El objetivo de esto es poder tener URLs semánticas para la identificación de los recursos.
- Sistema de plantillas. Se encarga de desplegar de manera dinámica las páginas web. Este genera HTML así como código Javascript basado en el código escrito en las plantillas.
- Internacionalización. Permite manejar varios lenguajes traduciendo solo las palabras necesarias.
- Estructura de almacenamiento en el caché que evita la regeneración de contenido que no ha cambiado.
- Extensión espacial *GeoDjango* para trabajar con bases de datos geoespaciales, la cual es una extensión geográfica ORM (*Object Relational Mapper*) que permite interactuar con extensiones espaciales de la base de datos.

Una ventaja de usar esta arquitectura es que los componentes tienen un bajo nivel de acoplamiento. Cada componente distinto de una aplicación hecha con la herramienta de desarrollo *Django*(se explicará mas adelante) tiene un propósito clave y puede ser cambiado de forma independiente sin afectar a otros componentes.

3.1.4. GeoNode

GeoNode está hecho con el *framework* Django, los componentes que lo integran son FOSS4G (Free and Open Source Software for Geospatial) como Geoserver[7], OpenLayers[8] , Ext JS[9], GeoExt[10], Pycsw[11] y como sistema manejador de base de datos [12] PostgreSQL con PostGIS como extensión para el manejo de datos geoespaciales [13]. Cada uno de estos componentes se explicarán mas adelante.

Fue diseñado para el trabajo colaborativo. Permite a múltiples usuarios subir datos de tipo *raster*, vectoriales y tablas con datos en formatos como *ESRI Shapefile*, GeoTIFF, KML y CVS (comma value separated). Las capas pueden ser editadas gráficamente, crear mapas con varias capas y ser compartidos. Gracias a la integración del sistema de administración de *Django* con el sistema de seguridad de GeoServer es posible asignar permisos sobre las capas[14].

[7]http://geoserver.org/
[8]http://openlayers.org/
[9]https://www.sencha.com
[10]http://www.geoext.org/
[11]http://pycsw.org/
[12] http://www.postgresql.org
[13]http://postgis.net
[14]Marco Minghini, *op. cit.*, p. 77

En la Figura 3.3, se divide en tres partes los componentes de GeoNode. En la primera se encuentran los encargados de mostrar los datos al usuario en forma de mapas interactivos en el navegador web. Estos se conectan a GeoNode en internet usando protocolos OGC. En la segunda parte del esquema, se encuentran los catálogos geoespaciales, django y GeoServer. La última esta compuesta por los componentes que se encargan de la persistencia a los datos como PosgreSQL o ArcGis Server.

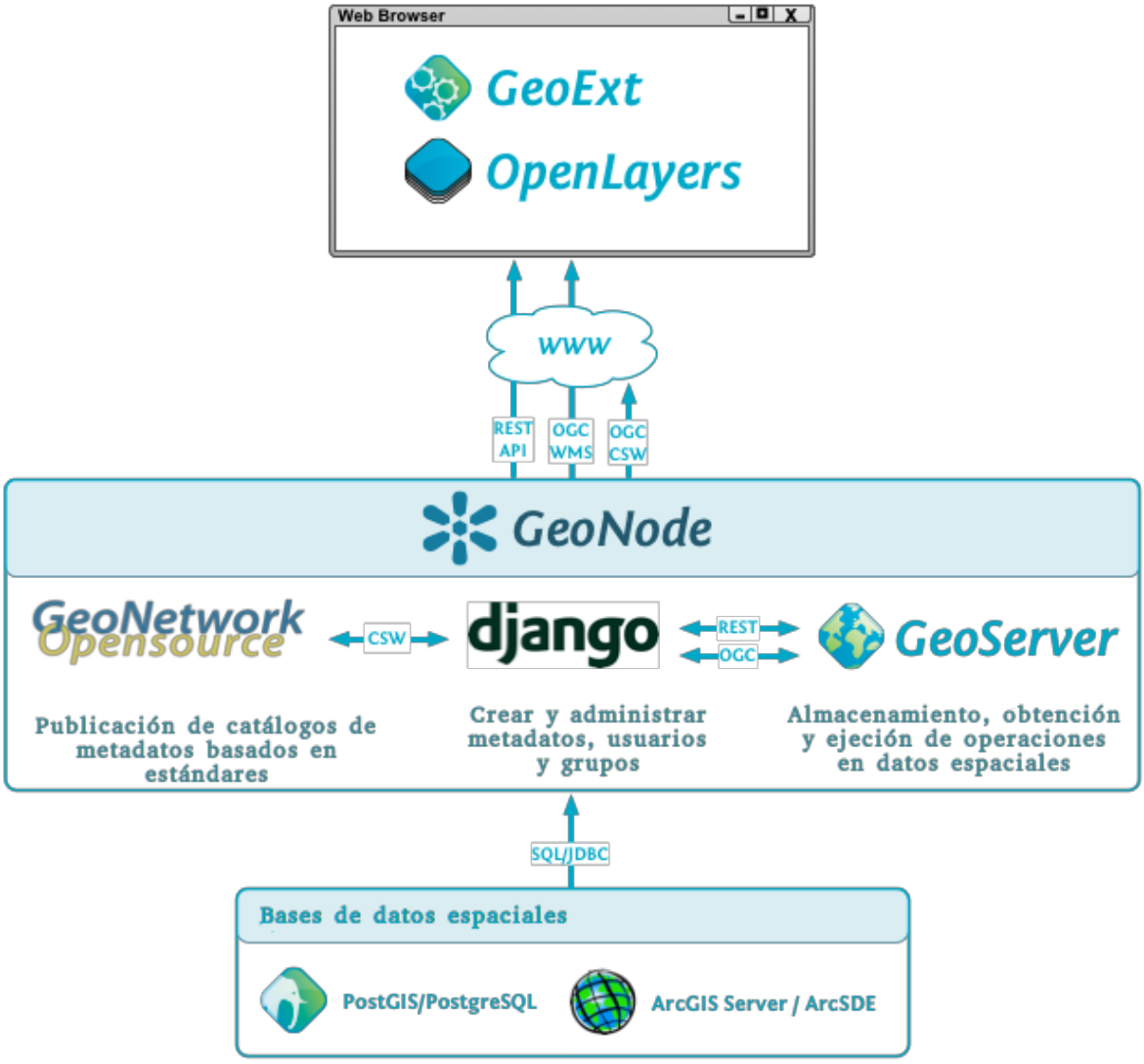


Figura 3.3: Principales componentes de GeoNode

Fuente: http://geonode.readthedocs.org/en/latest/reference/architecture.html

3.1.5. Apps de GeoNode

La interfaz de usuario de GeoNode está construida por *apps* que son módulos o aplicaciones reusables que están desarrolladas con *Django*. Por ejemplo, adaptar los elementos de las plantillas a nuestras necesidades. A continuación se mencionan y describen algunas

de las *apps* de GeoNode[44].

- *Base*, en esta *app* se almacenan las funcionalidades centrales usadas en GeoNode.
- *Documents*, permite la creación y administración de documentos que pueden estar relacionados con los mapas o capas.
- *Layers*, provee soporte para la administración y manipulación de conjuntos de datos geoespaciales llamados capas.
- *Maps*, permite la creación y administración de datos geoespaciales. Esta *app* de *Django* brinda la posibilidad de manipular conjuntos de datos geoespaciales. En particular tiene herramientas para editar, ver y buscar metadatos de mapas.
- *Proxy*, ayuda a aplicaciones*JavaScript* a acceder a servidores remotos. Esta textitapp provee algunos *proxies*. Un *proxy* es un servidor que sirve de intermediario entre un explorador web e Internet. Sirven para mejorar el rendimiento de una red, ya que almacenan una copia de las páginas web más utilizadas [45]. Estos son usados para acceder a los datos mediante servidores remotos para superar las restricciones impuestas por las políticas usadas por los buscadores.
- *Search*, provee mecanismos rápidos de búsqueda como por ejemplo búsqueda por etiquetas, metadatos, *bounding box*.
- *Security*, provee mecanismos de autentificación para asignar permisos a objetos.

Estas *apps* tienen una configuración por defecto, para personalizar GeoNode es necesario ajustar la configración de estas para propósitos específicos[15].

3.1.6. Arquitectura

La arquitectura de GeoNode se basa en un conjunto de herramientas básicas y bibliotecas que proveen las piezas para para construir a la aplicación. Para modificar GeoNode se debe tener un conocimiento básico de cada uno de estos componentes[46]. En la Figura 3.4 se muestra la arquitectura de GeoNode. Los rectángulos en color azul son las *apps* que lo componen, cada uno de estas se encarga de conectarse con otro software para dar una funcionalidad al sistema. Los que están en color naranja, con la ayuda de GeoExplorer se encargan de el despliegue de los mapas interactivos en el navegador web. Los que están debajo de el rectángulo de GeoNode son las bibliotecas geoespaciales que son usadas por las *apps*. En la parte inferior se encuentra GeoServer conectado a sus componentes.

[15] *Ibídem*

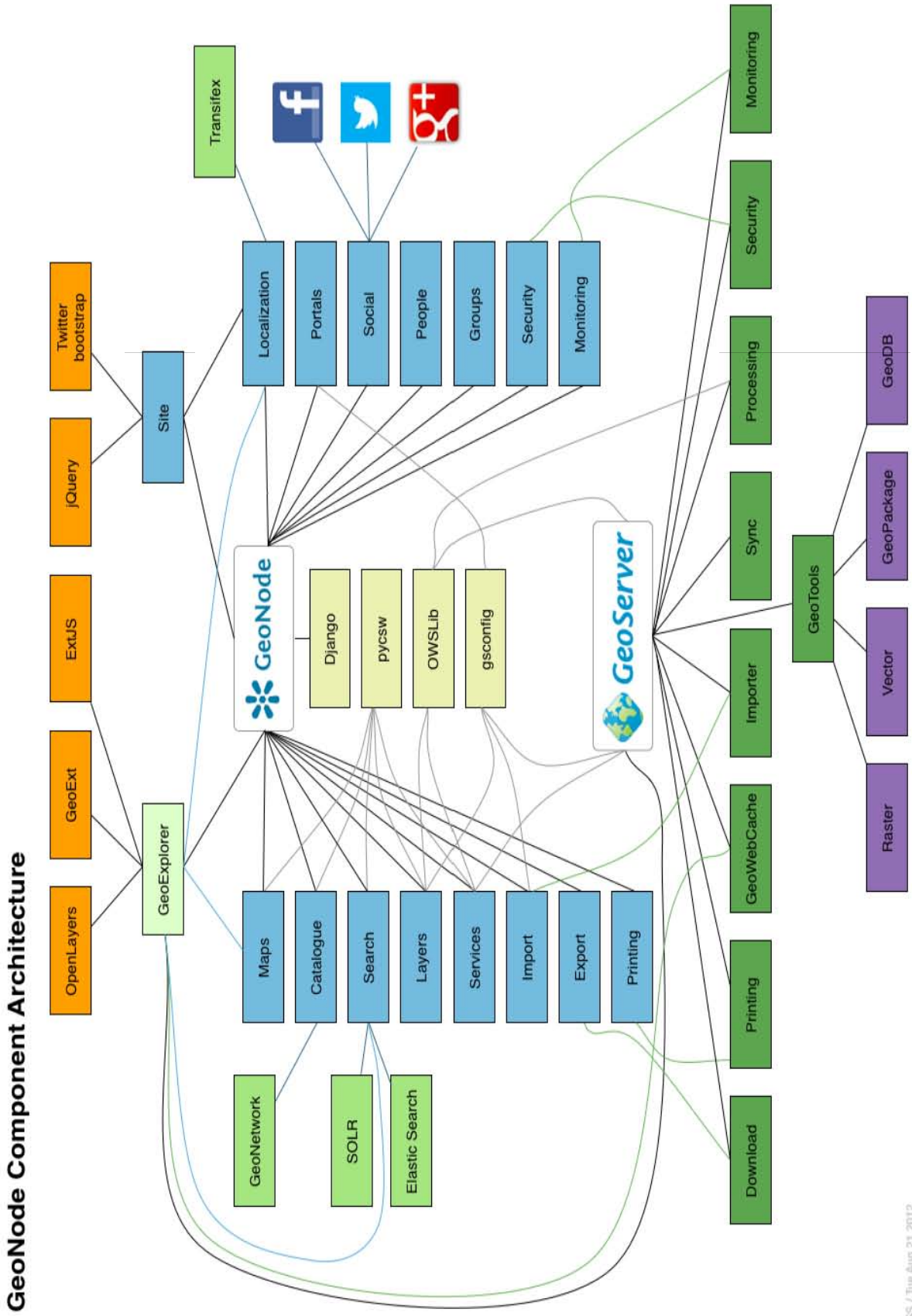


Figura 3.4: Arquitectura de GeoNode

Fuente: http://docs.geonode.org/en/dev/developers/architecture.html

3.1.7. GeoServer

Es una plataforma web para publicar datos geoespaciales, fue creado en el 2001 por la organización OpenGeo (recientemente fue renombrada como *Boundless*). Además de *Boundless*, el desarrollo también está a cargo de la compañía italiana GeoSolution y la empresa *Canadian Refraction Research*. Siendo un proyecto de la OsGeo, GeoServer se basa en la comunidad de todo el mundo de usuarios y desarrolladores. Escrito en Java, es una plataforma independiente y publicada bajo una licencia GNU GLP (*General Public License*)[16]. La versión actual es la 2.8, liberada el 29 de Mayo de 2015.

GeoServer provee una implementación de los protocolos WMS, WFS incluyendo las operaciones *Transaction* y *LockFeature*. Incluye un visualizador hecho en AJAX (*Asynchronous JavaScript And XML*) basado en OpenLayers para permitir la visualización de los datos. La interfaz de usuario se encuentra disponible en varios idiomas, provee herramientas de configuración que funcionan con documentos XML. Entre los aspectos que se pueden personalizar están la seguridad y los usuarios, espacios de trabajo(*Workspaces*), capas y grupos de capas, el almacenamiento de los datos (*stores*), servicios, estilos, proyecciones, *plugins* y el caché del servidor [17].

GeoServer es capaz de leer datos en formatos vectorial, *raster* y tiene un soporte maduro para bases de datos espaciales. También provee soporte completo para SLD (Styled Layer Descriptor) y múltiples formatos de salida incluyendo KML (Keyhole Markup Language) para una fácil integración con *Google Maps* y *Google Earth* [18].

Está diseñado para permitir la interoperabilidad y en él se pueden publicar datos provenientes de cualquier fuente importante utilizando estándares de la OGC [19]. En la Figura 3.5 se resumen las características de GeoServer. En la izquierda se muestra el manejo de datos vectoriales, los sistemas manejadores de bases de datos que soporta, los servidores de datos geoespaciales con los que puede conectarse y el soporte de formatos para datos *raster*. De lado derecho están los servicios OGC que implementa y los formatos de salida.

[16] Marco Minghini, *op. cit.*, p. 69
[17] *Ibídem*
[18] *Ibídem*
[19] Gregory Giuliani, *op. cit.*, p. 37

Figura 3.5: Servicios web y tipos de datos soportados por GeoServer

Fuente: http://boundlessgeo.com/solutions/solutions-software/geoserver/

3.1.8. GeoExplorer

GeoExplorer es una aplicación web basada en el *framework* GeoExt para componer y publicar mapas. Con GeoExplorer se pueden ensamblar mapas provenientes de GeoServer o cualquier servicio WMS e integrarlo con otros alojados en Google Maps y OpenStreetMap. También se puede editar información de los estilos, incrustar mapas compuestos en cualquier página web o desplegar los mapas en formato PDF [47].

Características de GeoExplorer[20]:

- Tiene una herramienta gráfica para dar estilo y editar;
- Permite subir *Shapefiles* y *GeoTIFFS*;
- Hace uso del almacenamiento en cache del servidor;
- Permite exportar mapas a pdf;
- Está construido con el cliente SDK de OpenGeo[21].

3.1.9. Apache

Es el servidor web HTTP más usado en el mundo, es ligero, altamente desarrollado y extremadamente modular. Este es el que se encarga de las peticiones de internet, respuestas y redirecciones con filtros y restricciones apropiadas. Posee una gran popularidad

[20] *Ibídem*
[21] *Ibídem*

y un desempeño certificado. Apache Tomcat también puede ser usado como un servidor estándar HTTP [22]. Apache HTTP server[23] es apoyado por una comunidad de desarrolladores bajo el mando de la Apache Software Foundation. Su primera versión fue liberada en el año de 1995 y desde 1996 ha sido el servidor web más popular en internet. Escrito en el lenguaje de programación C y distribuido bajo la licencia Apache 2.0, éste es usado mas comúnmente en sistemas tipo Unix, pero también se encuentra disponible para una amplia variedad de plataformas Linux, Windows y Mac OS X. La versión actual es la 2.4.12, liberada el 2015-01-29[24].

3.1.10. PostgreSQL

PostgreSQL es un sistema manejador de bases de datos desarrollado por *PostgreSQL Global Development Group*. Está escrito en el lenguaje C y puede ser ejecutado en la mayoría de las plataformas, es totalmente transaccional y obedece a los principios ACID (Atomicity, Consistency, Isolation, Durability), el conjunto de operaciones que garantizan que una base de datos es confiable. La concurrencia de transacciones es manejada por un sistema conocido como *Multiversion Concurrency Control* (MVCC), el cual permite cambios en la base de datos sin que los otros procesos los puedan ver hasta que dichos cambios se encuentren comprometidos citepostgres.

Entre las características de PostgreSQL se encuentran el soporte completo para llaves foráneas (*foreign keys*), uniones (*joins*), vistas (*views*), disparadores (*triggers*) y procedimientos almacenados en múltiples lenguajes (*stores prodedures*) como Java, Perl, Python, Ruby, C y C++, además de muchas bibliotecas; brinda soporte a la mayoría de tipos de datos SQL, también posee almacenamiento para grandes objetos en lenguaje binario incluyendo imágenes, sonido y vídeo. Permite usar conjuntos de caracteres internacionales (*multi-byte character encoding*), *unicode*,es sensible a las mayúsculas. PostgreSQL da soporte para varios índices funcionales entre los cuales se encuentran *B-tree*, *R-tree*, *hash* o árboles generalizados de búsqueda (Generalized Search Tree) (Hellerstein et al., 1995).

3.1.11. Bibliotecas geoespaciales de Python

GeoNode utiliza varias bibliotecas geoespaciales escritas en *Python* incluyendo *gsconfig* y *OSWLib*. *Gsconfig* se comunica con la API REST(*Representational State Transfer*) de configuración de GeoServer para configurar las capas de GeoNode en GeoServer. OWSLib se utiliza para tener conexión con los servicios OGC de GeoServer [48].

[22] Marco Minghini, *op. cit.*, p. 100
[23] http://httpd.apache.org
[24] *Ibídem*

3.1.12. jQuery

JQuery es una biblioteca de JavaScript que facilita manipular documentos HTML, el DOM (*Document Object Model*) de una página web, manejar eventos y crear animaciones. Es la biblioteca JavaScript más popular y es compatible con una amplia variedad de navegadores web, es gratuita y de código abierto [49].

GeoNode provee al usuario una interfaz interactiva y responsiva para esto usa varios *plugins* hechos con jQuery para proveer funcionalidad a ciertos elementos[46].

3.1.13. Bootstrap

Bootstrap es un *framework* de código abierto que permite el desarrollo web de forma rápida y fácil. Incluye plantillas basadas en HTML Y CSS (*Cascading Style Sheets)*, tipografía, formularios,tablas, botones, carruseles, e imágenes y barras de navegación. Bootstrap fue desarrollado por los programadores Mark Otto and Jacob Thornton en Twitter, y liberado como un producto open source en Agosto de 2011 en GitHub[50].

Las ventajas de usar Bootsrap son[25]:

- Fácil de usar: Cualquiera con conocimientos básicos de HTML y CSS puede usarlo.
- Comportamiento adaptativo o adaptable: Permite que las páginas se ajuste al tamaño de pantalla de dispositivos cliente heterogéneos como celulares, tabletas y monitores.
- Bootstrap es compatible con todas los navegadores modernos(Chrome, Firefox, Internet Explorer, Safari, and Opera).

3.1.14. Ext JS

Ext JS es una poderosa biblioteca de *JavaScript* para desarrollar aplicaciones web interactivas[51]. Es desarrollado por la compañía estadounidense Sencha y es compatible con la mayoría de los navegadores web y sistemas operativos. Ext JS está liberado bajo una estructura de licencia dual[26]. La principal ventaja de Ext JS es que se deriva de la amplia variedad de [27].

Alguna de las ventajas de Ext JS

- *Widgets* con interfaz de usuario personalizables;
- Componentes apropiadamente diseñados;

[25] *Ibídem*
[26] La version actual de Ext JS es la 4.2.2.
[27] *Ibídem*

- Documentación y tutoriales disponibles en la página de *Sencha*;
- La creación de ventanas, paneles o botones es intuitiva y simplificada así como la interacción con el DOM.

3.1.15. GeoExt

GeoExt es un *framework* escrito en *JavaScript* para hacer aplicaciones enriquecidas de mapeo web que combina las funcionalidades de *openlayers* con la interfaz de usuario de la biblioteca *ExtJS*. proporcionada por la compañía *Sencha*[52].

GeoExt reúne el modelo geográfico de OpenLayers, no solo es la conexión con OpenLayers, sino que permite crear nuevas funcionalidades y *widgets* .

Sus características son las siguientes[28]:

- Permite crear leyendas para los mapas;
- Tiene una administrador de opacidad para las capas;
- Incluye ventanas emergentes;
- Permite la administración de capas;
- Tiene un lector de *OGC capabilities*;
- Muestra las *capabilities*;
- Cuenta con formularios y herramientas de búsqueda.

La Figura 3.6 se pueden ver las ventanas y administración de capas hechas con GeoExt y OpenLayers.

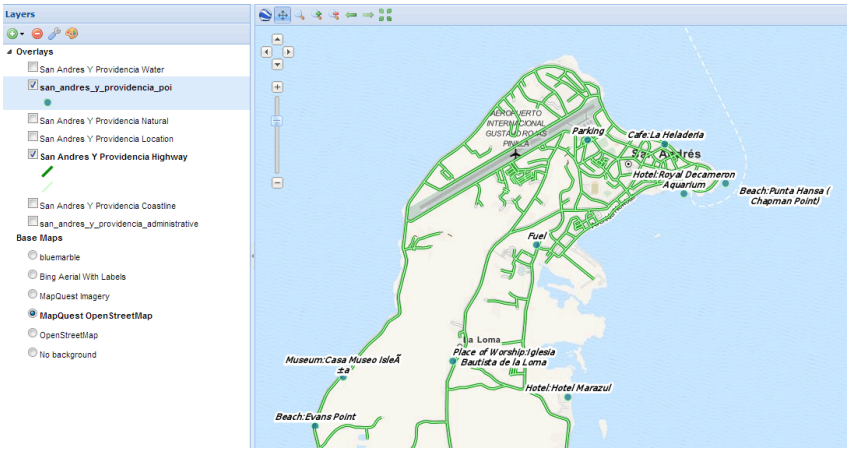


Figura 3.6: Ejemplo de un mapa usando GeoExt

Fuente: http://live.osgeo.org/en/quickstart/geonode_quickstart.html

[28]Simone Dalmasso, *op. cit.*, p. 63

3.2. Análisis de Requerimientos

3.2.1. Objetivo

Generar una versión digital del atlas expuesto en la introducción, que mantenga la calidad visual de la versión impresa, lo enriquezca agregando elementos de visualización interactiva y extender la audiencia del producto ofreciendo los recursos a través de servicios Web interoperables. Como punto de partida se cuenta con el atlas impreso "La Región del Volcán Chichón, Chiapas: Un espacio potencial para su protección, conservación y desarrollo sustentable". Lo que se desea es ajustar y extender a GeoNode utilizando el concepto de atlas. El atlas deberá estar organizado en secciones generales llamadas temas generales y estos a su vez divididos en subsecciones, que se llamarán temas específicos[29].

3.2.2. Requerimientos funcionales

Los requerimientos funcionales hacen referencia a las actividades y servicios que el sistema debe de proveer. Estos requerimientos están relacionados a las entradas que recibe el sistema, los procesos, salidas y los datos que se almacenan en el mismo [53].

Para implementar la funcionalidad de un atlas, se extenderá GeoNode incluyendo las siguientes secciones:

1. Página de inicio. Aquí se describen los objetivos, alcances y estructura del atlas.
2. Índice de temas, que se organiza en temas generales o secciones, que agrupan temas específicos. Estos corresponden a cada una de las que componen las 5 partes del atlas impreso a las que les llamaremos temas generales. Por ejemplo ejemplo en el caso del atlas del Chichón estos son: I. Naturaleza y ambiente, II. Climatología, III. Geomorfometría IV. Amenazas de origen natural y socioantrópico y V. Vulnerabilidad y riesgo. Las 5 secciones del atlas estará asociado a un color. En esta parte también se encuentra un listado de los temas específicos del atlas. Por cada elemento de la lista se mostrará su nombre, el tema de la hoja, los mapas que integran esa hoja, una pequeña imagen de muestra(*thumbnail*), una breve descripción y ligas hacia los temas específicos del atlas con sus respectivos mapas. Tendrá herramientas para buscar y descubrir el contenido del atlas.
3. Tema específico. Utilizamos el término "tema específico"para denotar cada uno de los tópicos desarrollados en extenso que forman parte de un tema general. Por ejemplo tema general "I. Naturaleza y Ambiente", es desarrollado en los subtemas "I.3 Topgrafía", "I.4 Espacio-mapa SPOT", "I.5 Geología", etc. Cada uno de estos consta de al menos un mapa, texto (que contiene el desarrollo en extenso del tema), una leyenda y algunos cuentan con imágenes o gráficas.

[29]Javier Osorno, *op. cit.*, p. 1

4. Carga de los datos. El sistema deberá proveer funcionalidades que permitan 1) cargar los temas generales y específicos, 2) cargar los elementos que componen cada uno de los temas específicos. Está dirigido al administrador o encargado de subir los datos al sistema.

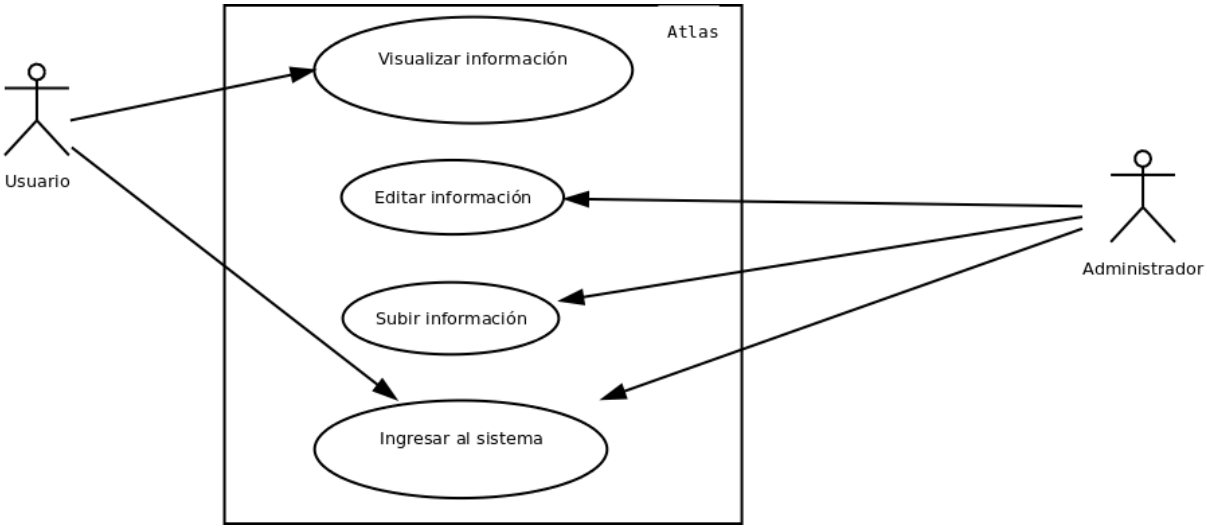


Figura 3.7: Diagrama de casos de uso General

3.2.3. Requerimientos no funcionales

Los requerimientos no funcionales describen características que debe tener el sistema para tener éxito como rendimiento, usabilidad, costo, tiempo de entrega y seguridad [30].

Los requerimiento no funcionales son los siguientes[31]:

1. En general el sistema debe extender a GeoNode integrando el concepto de atlas.
2. Enriquecerá la calidad del producto agregando elementos de visualización interactiva.
3. Extenderá la audiencia y uso a través de servicios web interoperables de la OGC.
4. El atlas debe poder ser visualizado desde el explorador web Mozilla o el navegador web Chrome, sin importar el sistema operativo que el usuario utilice.
5. Por tratarse de una página web, el atlas debe estar en un servidor que responda a las peticiones de cualquier usuario sin importar en que punto geográfico se encuentre.
6. Tendrá que usar software de código abierto con el fin de evitar costos económicos.
7. Deberá mantener la calidad gráfica de los contenidos del atlas impreso (mapas, ilustraciones, tablas, etc).

[30] VicenÇFernández, *op. cit.*, p. 85
[31] Javier Osorno, *op. cit.*, p. 1

8. Tendrá mecanismos para facilitar la construcción colaborativa.

3.3. Diseño e implementación del atlas

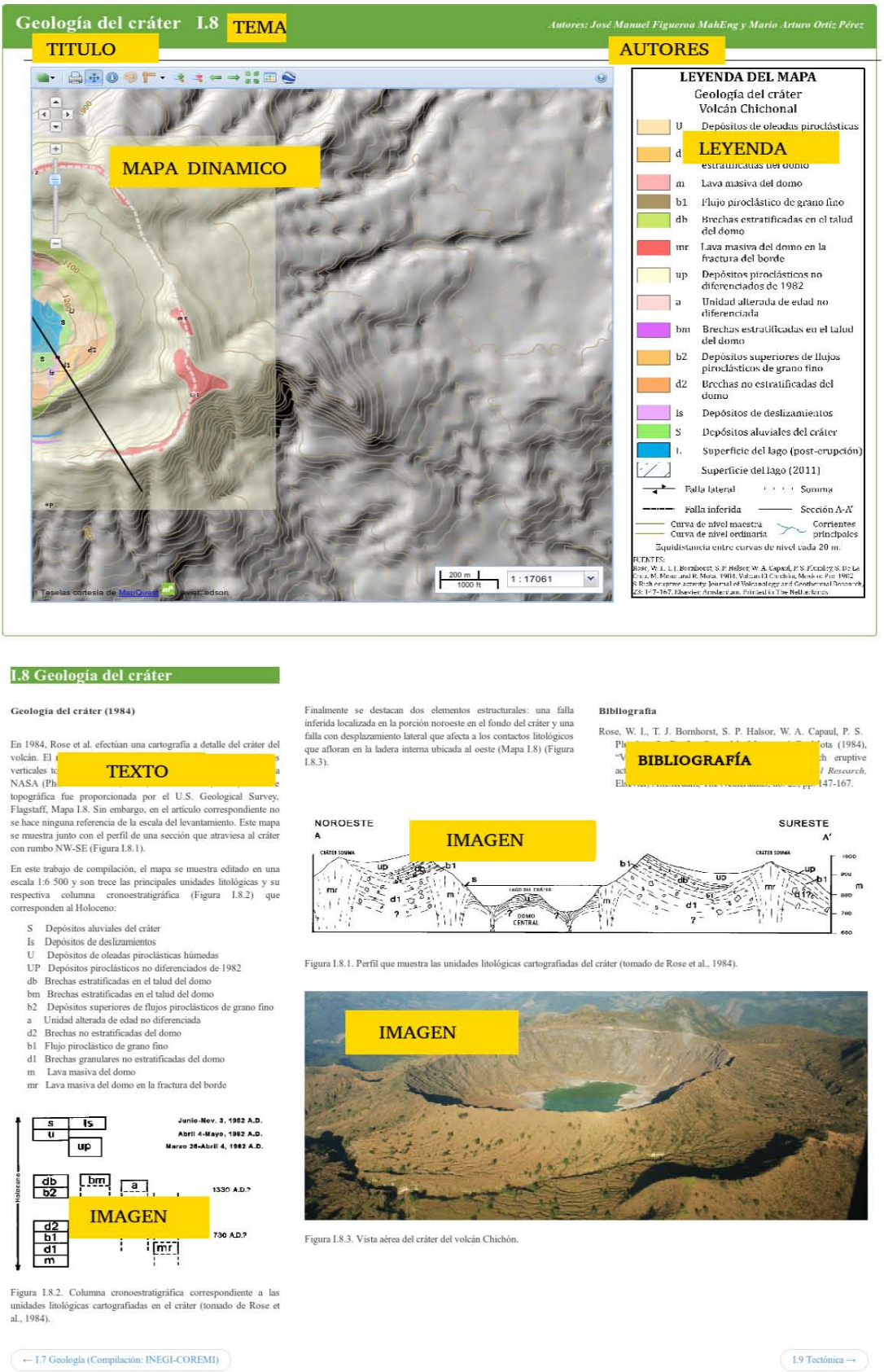
3.3.1. Prototipo inicial

Cabe mencionar que en el 2014 se generó una primer versión del atlas usando los mapas que ya se tenían cargados en GeoNode, pero no el *framework*. Se necesitaba mostrar el contenido del atlas para ser presentados en una ponencia de la Sociedad Latinoamericana de Percepción Remota y Sistemas de Información Espacial (SELPER)[4], pero el sistema aún no estaba listo. Entonces se generaron algunos temas específicos del atlas a mano, se editaron las hojas una por una, colocando los mapas, las fotos, las ilustraciones y el texto de forma manual. En está versión precedente al atlas, las hojas del atlas son páginas HTML estáticas, por lo cual se buscó mejorarla.

Para este proyecto se planteo el objetivo general de extender GeoNode para dar soporte al concepto de atlas. Esto requería:

- Leer la información de la base de datos. Extender el modelo de objetos de Geonode para habilitar el almacenamiento persistente de los componentes de un atlas.
- Extender el sistema de plantillas.
- Generar la parte de los controladores.
- Cargar los temas específicos faltantes.
- Corregir algunas ligas de la página.
- Acoplar debidamente los temas específicos a GeoNode.

En la Figura 3.8 se muestra un tema específico del prototipo inicial en el se señalan los elementos que lo componen.



I.8 Geología del cráter

Geología del cráter (1984)

En 1984, Rose et al. efectúan una cartografía a detalle del cráter del volcán. El [illegible] verticales tc [illegible] NASA (Ph [illegible] topográfica fue proporcionada por el U.S. Geological Survey, Flagstaff, Mapa I.8. Sin embargo, en el artículo correspondiente no se hace ninguna referencia de la escala del levantamiento. Este mapa se muestra junto con el perfil de una sección que atraviesa al cráter con rumbo NW-SE (Figura I.8.1).

TEXTO

En este trabajo de compilación, el mapa se muestra editado en una escala 1:6 500 y son trece las principales unidades litológicas y su respectiva columna cronoestratigráfica (Figura I.8.2) que corresponden al Holoceno:

- S Depósitos aluviales del cráter
- ls Depósitos de deslizamientos
- U Depósitos de oleadas piroclásticas húmedas
- UP Depósitos piroclásticos no diferenciados de 1982
- db Brechas estratificadas en el talud del domo
- bm Brechas estratificadas en el talud del domo
- b2 Depósitos superiores de flujos piroclásticos de grano fino
- a Unidad alterada de edad no diferenciada
- d2 Brechas no estratificadas del domo
- b1 Flujo piroclástico de grano fino
- d1 Brechas granulares no estratificadas del domo
- m Lava masiva del domo
- mr Lava masiva del domo en la fractura del borde

Figura I.8.2. Columna cronoestratigráfica correspondiente a las unidades litológicas cartografiadas en el cráter (tomado de Rose et al., 1984).

Finalmente se destacan dos elementos estructurales: una falla inferida localizada en la porción noroeste en el fondo del cráter y una falla con desplazamiento lateral que afecta a los contactos litológicos que afloran en la ladera interna ubicada al oeste (Mapa I.8) (Figura I.8.3).

Bibliografía

Rose, W. I., T. J. Bornhorst, S. P. Halsor, W. A. Capaul, P. S. [illegible] ota (1984), "[illegible] eruptive act[illegible] *Research*, Elsev[illegible] 147-167.

BIBLIOGRAFÍA

Figura I.8.1. Perfil que muestra las unidades litológicas cartografiadas del cráter (tomado de Rose et al., 1984).

Figura I.8.3. Vista aérea del cráter del volcán Chichón.

← I.7 Geología (Compilación: INEGI-COREMI)
I.9 Tectónica →

Figura 3.8: Aspecto de la versión anterior.

Fuente: Prototipo inicial

3.3.2. Arquitectura del sistema

La arquitectura del atlas cumple con las características del patrón *Model-View-Template* (MVT), ya que es la estructura que siguen los sistemas hechos con Django. Al crear la *app*, el *framework* se encarga de generar los archivos básicos para codificar el *Modelo*, la *Vista* y el *Template* o plantilla.

Los archivos son los siguientes:

- El archivo *models.py* contiene la definición de las tablas del modelo de datos del atlas. Esta es la parte del modelo de la aplicación. El siguiente es un ejemplo de la definición de un modelo de la base de datos. Esta clase define un teme especifico del atlas.

```
class Hoja_atlas(models.Model):
    fecha = models.DateTimeField('Fecha de publicacion')
    autor = models.ManyToManyField(Autor, null=True)
    mapa = models.ManyToManyField(Map)
    atlas = models.ForeignKey(Atlas)
    tema = models.CharField(max_length=10)
    nombre = models.CharField(max_length=200)
    texto = models.TextField()
    color = models.CharField(default='#49A8C5',max_length=7)
    thumbnail = models.FileField(default='/static/geonode/
        img/missing_thumb.png',upload_to='Thumbnail/',
                                    null=True,
                                    blank=True,
                                    verbose_name='Thumbnail')
    descripcion = models.CharField(max_length=50)
    informacion_bibliografica = models.TextField()
    def __unicode__(self):
        return self.nombre
```

- El archivo *views.py* contiene la parte que se encarga de la lógica y comportamiento de la aplicación. Este es el controlador de la aplicación. El siguiente código es un ejemplo de una función. Esta obtiene todos los temas específicos y los envía a la plantilla que los desplegara.

```
def index(request):
    atlas = Atlas.objects.all()
    hojas = Hoja_atlas.objects.all()
    results = Map.objects.all()
    return render_to_response('atlas/index.html',
        RequestContext(request,{'hojas': hojas,'results':
            results,'atlas': atlas}))
```

- El archivo *urls.py* contiene los patrones de las URL y su relación con las funciones

definidas en el archivo *views.py*. El siguiente es un ejemplo de como se definen las urls. Aquí están algunas de las urls del atlas.

```
url(r'^atlas/(?P<hoja_atlas_id>[0-9]+)/$','atlas.views.hoja'
    ),
url(r'^atlas/$', 'atlas.views.index'),
url(r'^atlas/cargarAtlas$', 'atlas.views.cargarAtlas'),
url(r'^atlas/cargarHoja$', 'atlas.views.cargarHoja'),
url(r'^atlas/cargarGrafica$', 'atlas.views.cargarGrafica'),
```

- En la carpeta *templates* se almacenan los documentos HTML de la aplicación. Esta es la parte del *template*.

  El siguiente segmento de código ilustra como se forma una plantilla.

```
{% for hoja in hojas %}
<article id="{{ hoja.color }}">
<div class="content">
<div class="item-header">
<a href="{{ hoja.id }} "><img class="thumb" src="{{ hoja.
    thumbnail.url}}" /></a>
<h3><i class="icon-map-marker"></i> <a href="{{ hoja.id }}"
    >{{hoja.tema|capfirst}} {{hoja.nombre|
```

- En el archivo *admin.py* se dan de alta los modelos de la base de datos para poder crear, eliminar y editar objetos. El siguiente segmento de código muestra cómo se registran los modelos para manipularse desde la interfaz de administración.

```
from atlas.models import Atlas, Hoja_atlas, Autor, Tabla
from django.contrib import admin
    admin.site.register(Atlas)
    admin.site.register(Hoja_atlas)
    admin.site.register(Tabla)
    admin.site.register(Autor)
```

- El archivo *forms.py* guarda las formas para subir elementos al sistema. Este archivo es parte de la vista.

  El siguiente segmento de código es una definición de un formulario.

```
class AtlasForm(forms.ModelForm):
    class Meta:
        model = Atlas
```

- El archivo *init.py* sirve para indicar a *Python*, que este directorio debe ser considerado como un paquete de *Python*.

- En el archivo *tests.py* se guardan las pruebas de la aplicación.

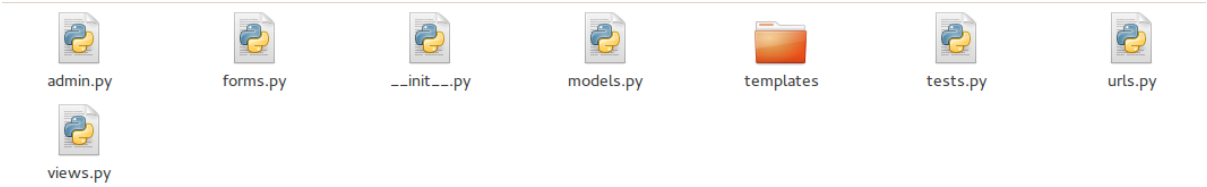


Figura 3.9: Archivos de la aplicación

3.3.3. Diseño del comportamiento

Se heredó el comportamiento de navegación, de formularios y el despliegue de datos estáticos con el que ya contaba GeoNode. En el siguiente diagrama se observa como los componentes MVT interactúan para llevar a cabo la acción de desplegar los datos de una sección del atlas, primero la parte de la Vista es activada por el usuario al ingresar a la sección, a continuación la vista solicita al controlador los datos del Modelo para ser desplegadas al usuario en la parte de la Vista.

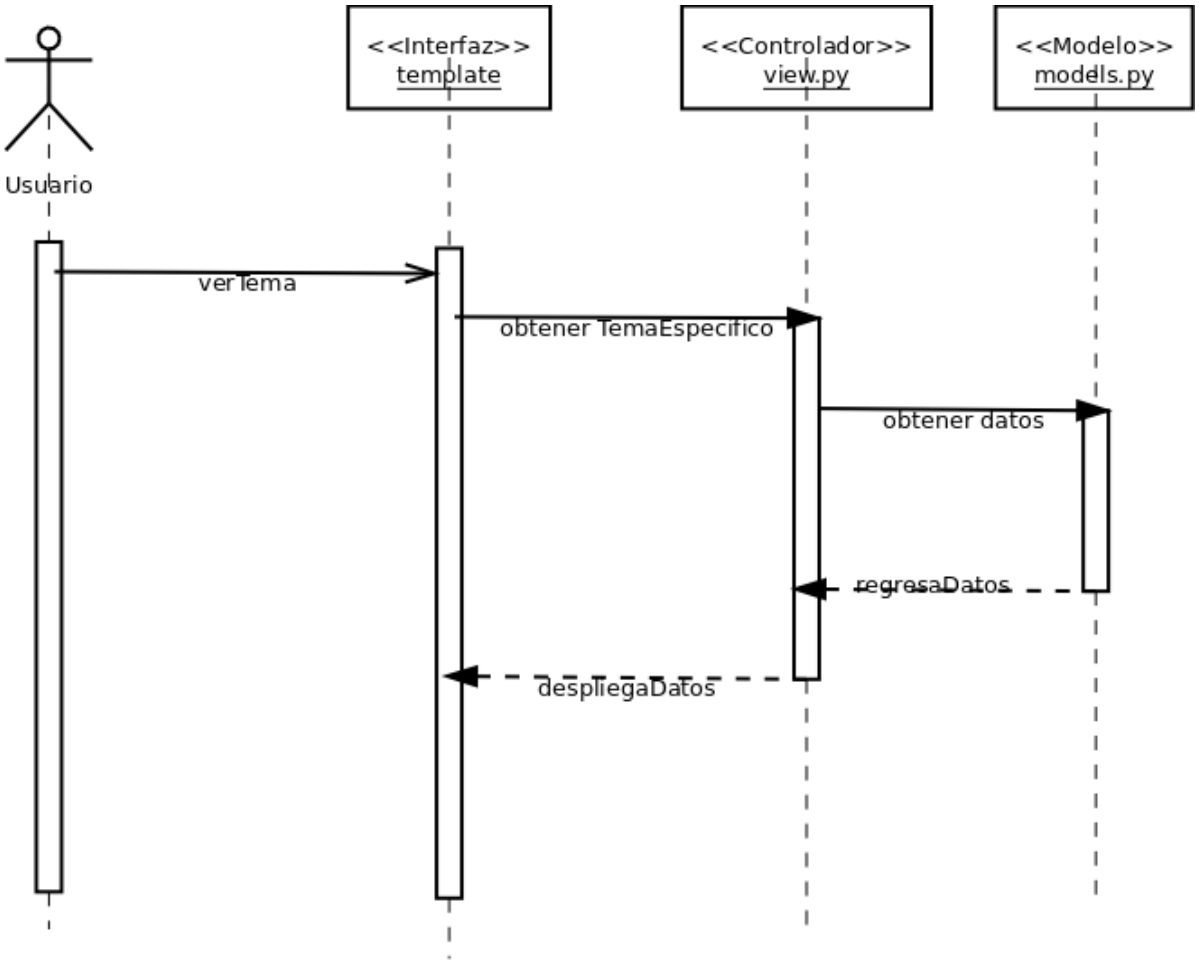


Figura 3.10: Diagrama de secuencia para solicitar un tema especifico

3.3.4. Diseño de la interfaz

Por simplicidad se decidió que el atlas tuviese la misma apariencia que GeoNode, ya que su interfaz es a la que el usuario está acostumbrado a usar. El diseño de la interfaz se hizo de tal forma que fuese sencillo de usar y que no hubiesen grandes diferencias entre GeoNode y el atlas. Para las hojas se retomó el diseño y colores del atlas anterior. Se hicieron prototipos de pantalla para la lista de temas, la hoja del atlas y el sistema de carga. Se incluye el mapa de navegación para complementar los prototipos.

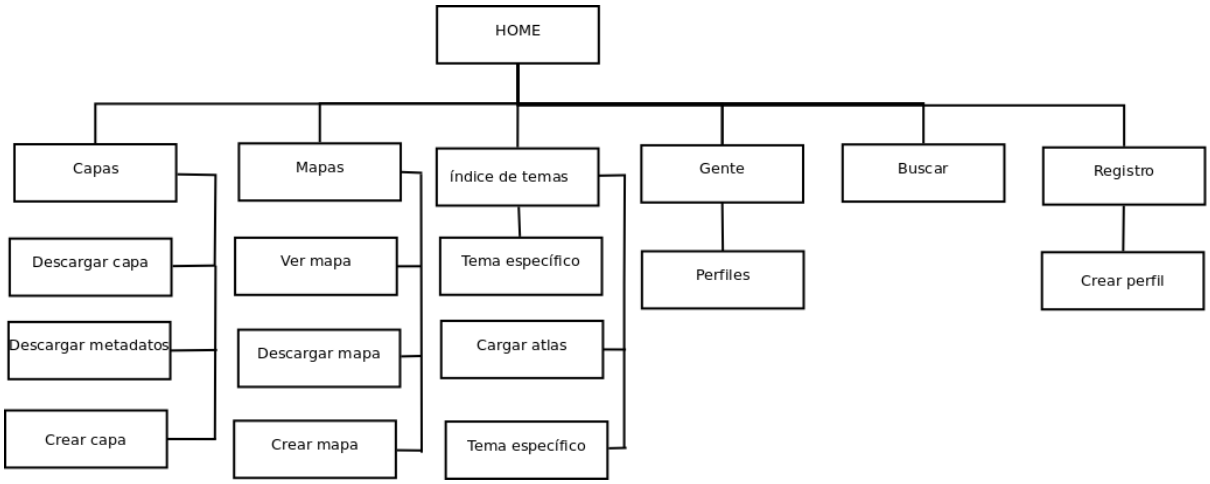


Figura 3.11: Mapa de navegación del sitio

La Figura 3.12 es el prototipo de pantalla para el índice de temas específicos. A continuación se explican los elementos que lo componen:

1. Es la barra superior de la página. Desde esta se puede acceder a secciones.
2. Son botones que sirven para llegar a las secciones de carga de datos.
3. Campos de búsqueda.
4. Es un elemento del índice de temas. Este contiene los datos más importantes del tema específico como nombre, tema, autores, pequeña imagen de muestra, una breve descripción, fecha de creación, puntuación y el número de visitas.

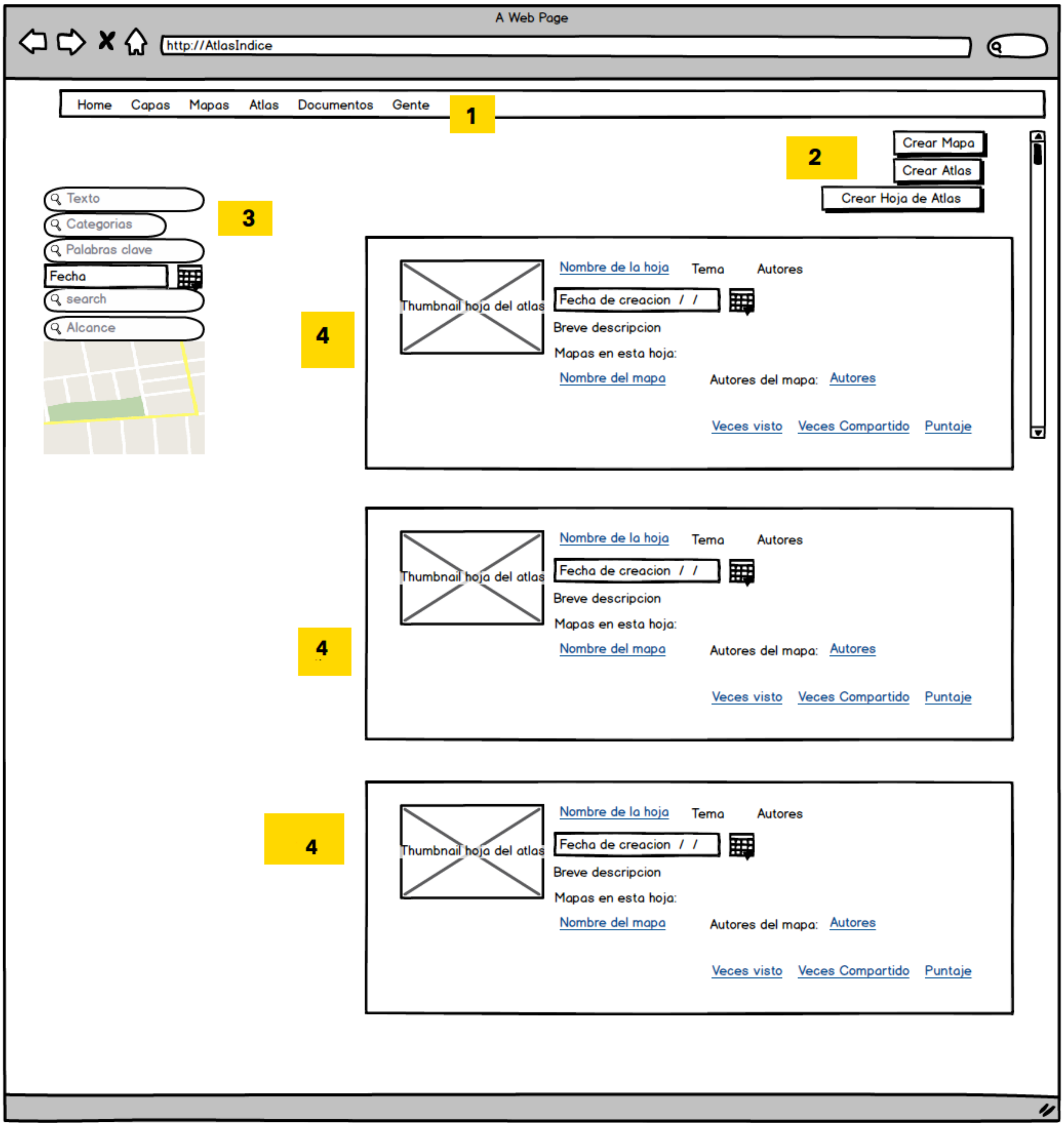


Figura 3.12: Prototipo de pantalla para la sección del índice de temas de atlas

La Figura 3.13 es el prototipo de pantalla para la sección de tema específico. En este se muestra la distribución y nombre de los componentes.

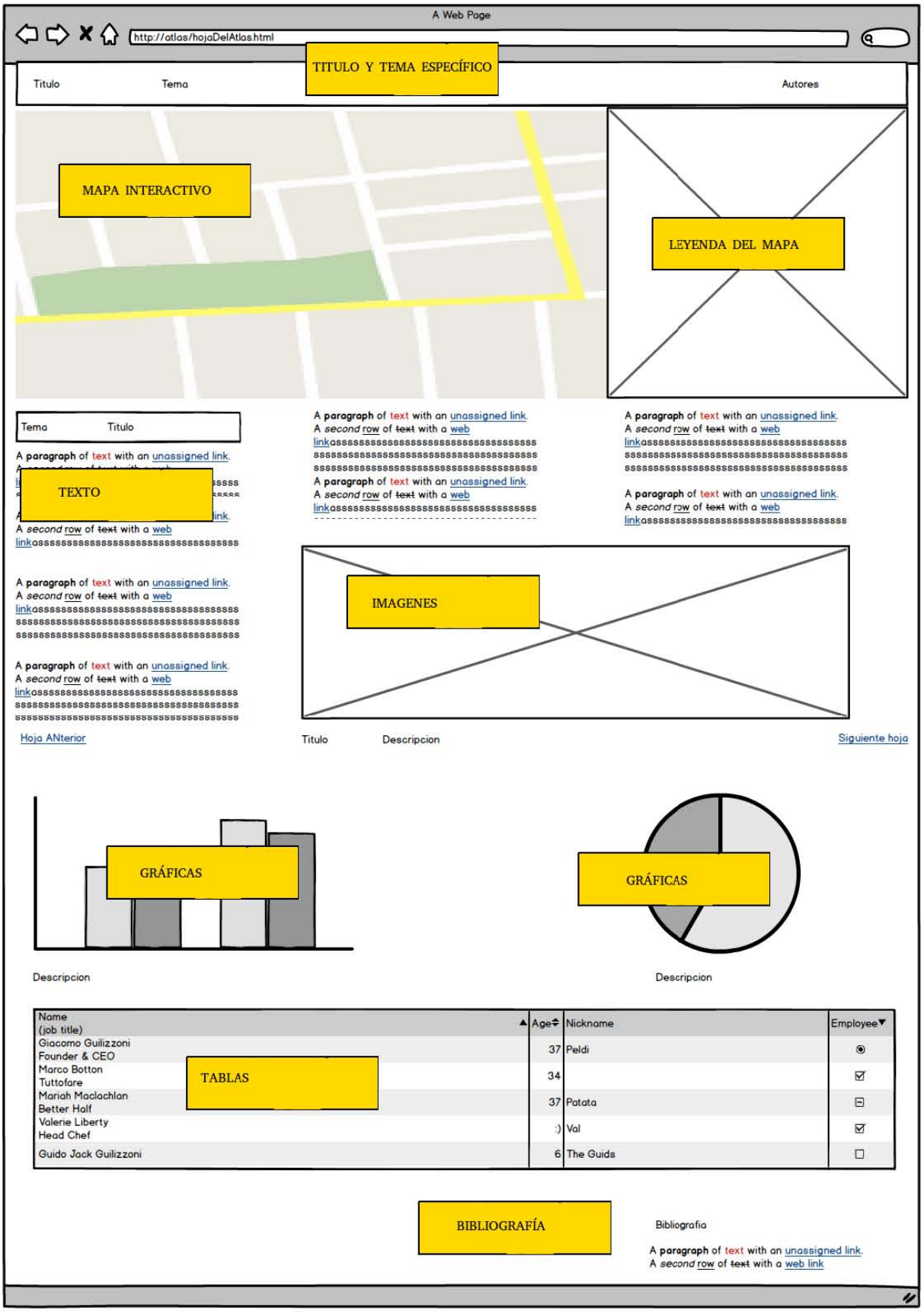


Figura 3.13: Prototipo de pantalla para la sección de los temas especifico

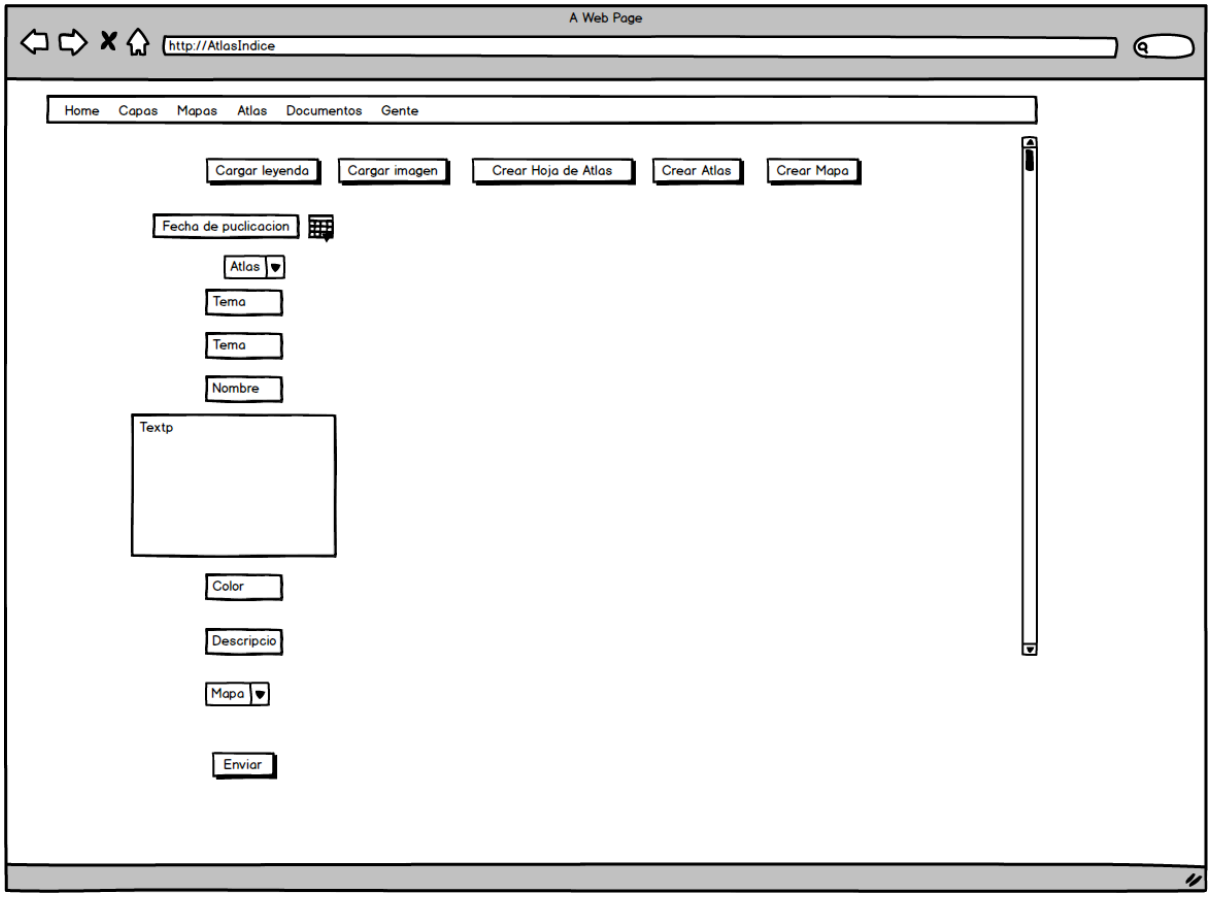


Figura 3.14: Prototipo de pantalla para la sección de la carga de datos

3.3.5. Diseño del esquema de datos

Para el diseño de la base de datos se reviso el atlas impreso y se listaron el tipo de elementos que contiene cada tema. Todos los temas cuentan con uno o varios mapas, lista de autores, titulo, número del tema, texto, leyenda del mapa y en ocasiones imágenes, bibliografía, gráficas o tablas.

El esquema de datos utilizado para el atlas, consta de ocho tablas con sus respectivas relaciones entre ellas, estas se muestran en el siguiente diagrama.

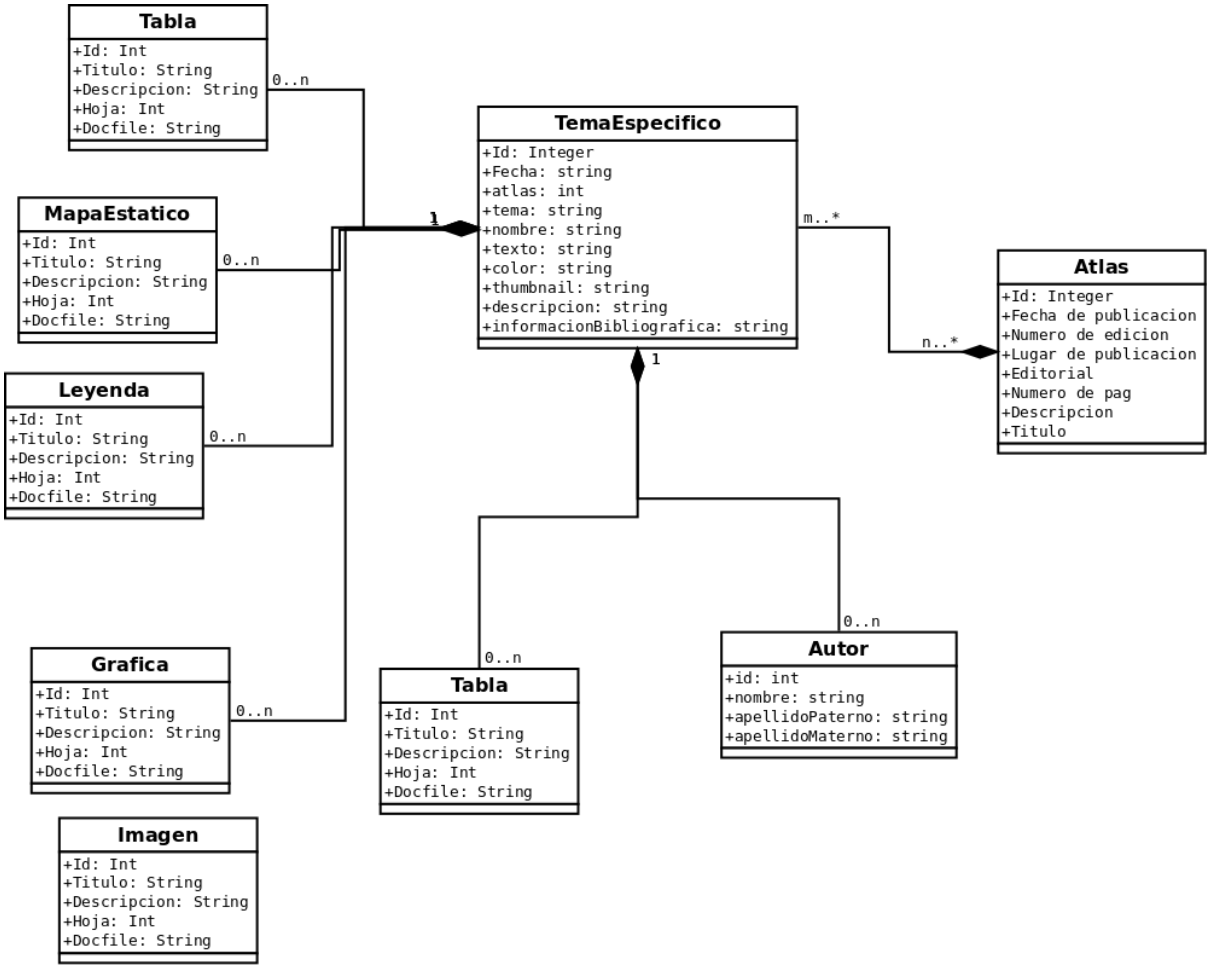


Figura 3.15: Diagrama de clases

En la **tabla Atlas** se definieron los campos para almacenar los datos más importantes de un atlas impreso como el título, fecha de publicación, número de edición, lugar de publicación, editorial, número de páginas y un atributo tipo texto para guardar una descripción general sobre el libro.

En la **tabla Leyenda** se definieron los atributos necesarios para almacenar la leyenda de cada mapa, que en este caso se guardaran en forma de imagen. Se encuentra compuesta por los campos título, descripción, y *docfile*, que sirve para asignar la ruta donde se almacenan los archivos.

En la **tabla Autor** se encuentran los campos necesarios para almacenar el nombre, apellido materno y paterno del o los autores de la hoja del atlas.

La tabla con el nombre **TemaEspecifico**, mostrada en la Figura , contiene diez campos. Los datos que componen a una hoja del atlas los cuales son la fecha en la que la hoja se registró en el sistema, el tema que le corresponde a lo hoja, el nombre que le pertenece, el texto que acompaña al mapa, el color que le corresponde a ese tema, un *thumbnail* de la hoja del atlas, una descripción corta que sirve para dar una breve introducción sobre la hoja y un campo para guardar la información bibliográfica. El

campo de mapa resulta importante porque sirve para relacionar los mapas a las hojas del atlas. Esta es la tabla principal en el modelo de la aplicación.

La tabla **MapaEstatico** sirve para almacenar las imágenes de los mapas(en caso de que no se cuente con el mapa interactivo). Para almacenarlas se tienen los campos de titulo, descripción y un campo para especificar la ruta del archivo.

La tabla **Imagen** se usa para almacenar imágenes del atlas, tiene campos para guardar el nombre, una descripción y un campo para la ubicación de la carpeta.

Las clase con el nombre **Tabla** sirve para almacenar información que viene forma de columnas y renglones del atlas, se pueden guardar en HTML para poder ser desplegadas en la plantilla o guardarse como imagen, también se pueden guarda el nombre, descripción y la ubicación del archivo.

La **tabla Grafica**, almacena gráficas en forma de imágenes o en código *JavaScript* que las hace interactivas.

3.3.6. Implementación

El diseño pretende que las clases sean lo más sencillas para ahorrar tiempo y esfuerzo al momento de implementarlas. A partir del diagrama de clases se generaron las tablas. Se usaron los prototipos de pantalla para realizar las vistas del a aplicación.

3.3.7. Integración con GeoNode

Los pasos para integrar la *app* con GeoNode fueron los siguientes:

1. Crear un *template* basado en el ejemplo del proyecto de geonode.

   ```
   $ django-admin startproject my_geonode --template=https://
     github.com/GeoNode/geonode-projectarchive2.0.zip -epy,rst
   $ sudo pip install -e my_geonode
   ```

2. Editar el archivo /etc/apache2/sites-available/geonode y cambiar la siguiente linea

   ```
   WSGIScriptAlias / /var/www/geonode/wsgi/geonode.wsgi por
     WSGIScriptAlias / /path/to/my_geonod emy_geonode/wsgi.py
   ```

3. Reiniciar apache:

   ```
   $ sudo service apache2 restart
   ```

4. Renombrar el archivo *local_settings.py.sample* a *local_ settings.py*

5. Copiar la carpeta con la *app* a la altura del archivo manage.py.

6. Agregar el nombre de la *app* a la lista de *apps* instaladas en el archivo settings.py

7. Copiar las URLs al archivo /usr/local/lib/python2.7/dist-packages/geonode/urls.py
8. Sincronizar la base de datos

```
python manage.py syncdb
```

3.3.8. Ambiente de desarrollo

Se virtualizó Ubuntu con las siguientes características: Ubuntu 12.04 LTS(*Long Term Support*) de 64 bits, 30 GB disco duro, 3 GB de memoria RAM, GeoNode 2.0. Lo anterior se hizo para poder hacer pruebas y en caso de que el sistema dejará de servir, deshacernos de esa virtualización y usar otra.

Debido a que Python es un lenguaje de programación que no requiere de herramientas especificas para programar, se utilizó lo siguiente:

- Terminal con el ambiente de desarrollo activo.
- Terminal con el servidor de desarrollo en ejecución.
- Editor de texto(en este caso usé *gedit*)
- La herramienta *Developer Tools* del navegador *Chromium*(para inspeccionar páginas y conexiones)

3.3.9. Carga del atlas

Algunas capas y mapas interactivos ya se encontraban cargadas en el servidor. Primero se creó un atlas para después crear hojas del atlas que lo componen. Esto se realizó en un servidor de pruebas. El cual cuenta con las siguientes :

- Ubuntu 12.04 LTS de 64 bits
- 6 GB de memoria RAM
- Procesador Intel Xeon E3-1220 v3 (3.10 GHz)
- 300 GB de disco duro para almacenamiento

Como sistema manejador de bases de datos se usa *Postgres*, debido a que tiene soporte para *PostGIS*, *Apache* como servidor por ser una herramienta de amplio uso.

Se cuenta con un pdf del atlas. Para cargar el mapa se copiaron todos los textos, fotografías, tablas, imágenes del atlas para poder cargar a mano las hojas del atlas una por una. Se reuso parte del trabajo del portal anterior, como el hecho de no tener que copiar todas las imágenes del atlas porque algunas ya se encontraban almacenadas en el servidor. El proceso se ilustra en la Figura 3.16.

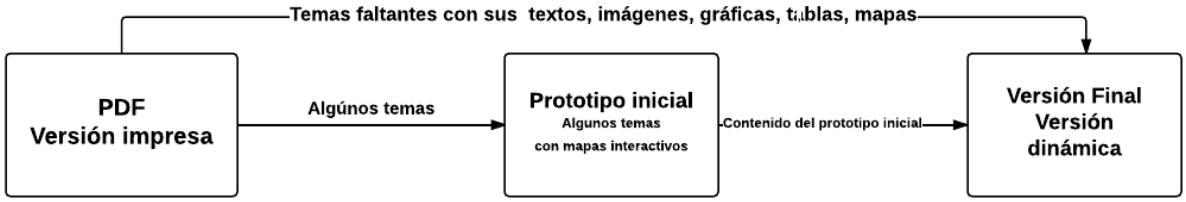


Figura 3.16: Secuencia entre las versiones del atlas

En total se almacenaron 64 temas específicos del atlas. Para cada uno de estos se cargaron los siguientes elementos:

- Tema de la hoja
- Titulo de la hoja
- Autores
- Texto de la hoja
- Mapas
- Mapas estáticos(cuando no se tenia el mapa interactivo)
- Imágenes
- Citas bibliográficas
- Gráficas(en algunos casos)
- Tablas (en algunos casos)

3.3.10. Vista final

A continuación se muestran impresiones de pantalla del atlas en línea. La Figura 3.17 es la pantalla de inicio del atlas. Aquí se muestran imágenes del cráter e información general sobre el contenido del atlas(numero de capas, usuarios, mapas). El mapa de navegación es el mismo que el de la Figura 3.11.

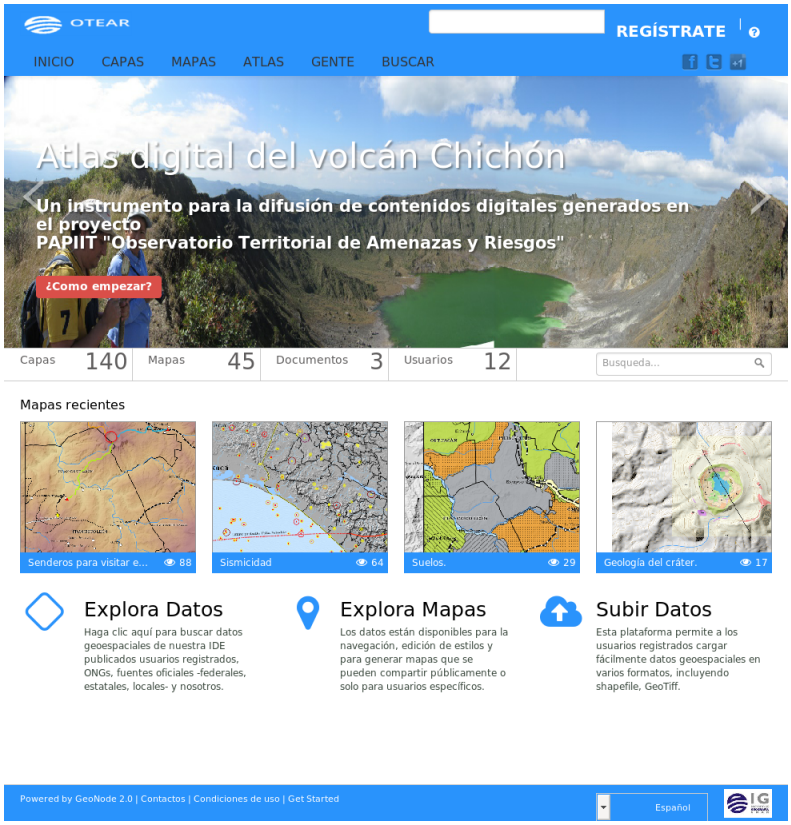


Figura 3.17: Página incial

En la Figura 3.18 se muestra la sección donde se listan los temas específicos. En está es posible acceder a una lista con los temas disponibles o realizar una búsqueda.

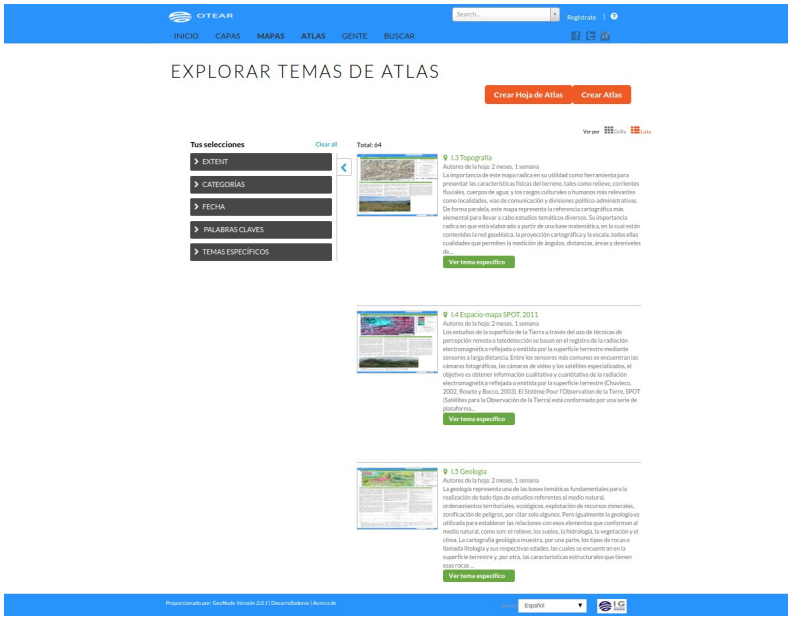
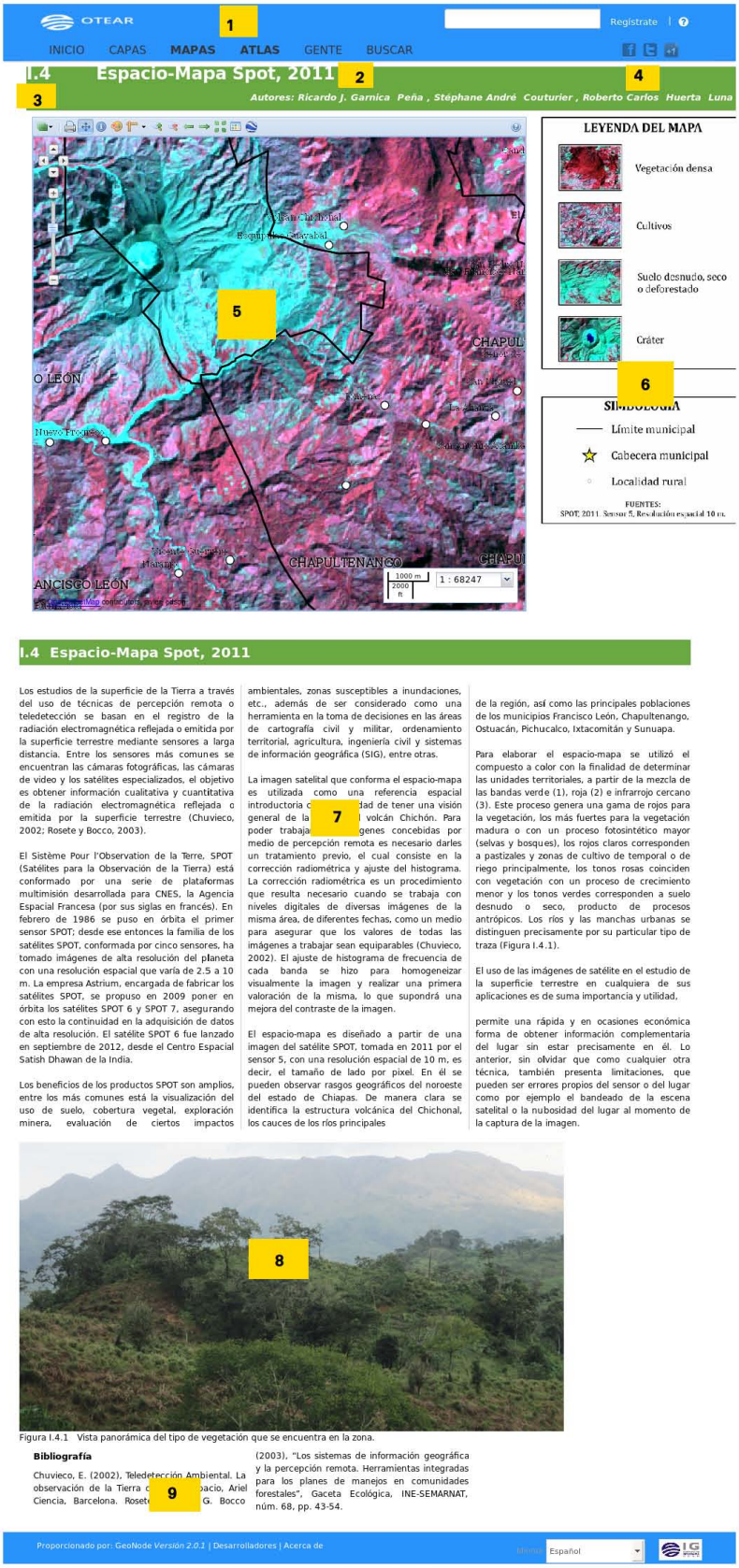


Figura 3.18: Índice de temas específicos

La Figura 3.19 es un ejemplo de un tema especifico. En este se despliegan todos los

elementos que lo conforman. Los elementos son:

1. Cabecera de la página.
2. Nombre del tema específico.
3. Tema.
4. Titulo del tema específico
5. Autores.
6. Mapa interactivo.
7. Leyenda del mapa.
8. Texto del tema específico.
9. Imagen.
10. Bibliografía.



I.4 Espacio-Mapa Spot, 2011

Los estudios de la superficie de la Tierra a través del uso de técnicas de percepción remota o teledetección se basan en el registro de la radiación electromagnética reflejada o emitida por la superficie terrestre mediante sensores a larga distancia. Entre los sensores más comunes se encuentran las cámaras fotográficas, las cámaras de video y los satélites especializados, el objetivo es obtener información cualitativa y cuantitativa de la radiación electromagnética reflejada o emitida por la superficie terrestre (Chuvieco, 2002; Rosete y Bocco, 2003).

El Sistème Pour l'Observation de la Terre, SPOT (Satélites para la Observación de la Tierra) está conformado por una serie de plataformas multimisión desarrollada para CNES, la Agencia Espacial Francesa (por sus siglas en francés). En febrero de 1986 se puso en órbita el primer sensor SPOT; desde ese entonces la familia de los satélites SPOT, conformada por cinco sensores, ha tomado imágenes de alta resolución del planeta con una resolución espacial que varía de 2.5 a 10 m. La empresa Astrium, encargada de fabricar los satélites SPOT, se propuso en 2009 poner en órbita los satélites SPOT 6 y SPOT 7, asegurando con esto la continuidad en la adquisición de datos de alta resolución. El satélite SPOT 6 fue lanzado en septiembre de 2012, desde el Centro Espacial Satish Dhawan de la India.

Los beneficios de los productos SPOT son amplios, entre los más comunes está la visualización del uso de suelo, cobertura vegetal, exploración minera, evaluación de ciertos impactos ambientales, zonas susceptibles a inundaciones, etc., además de ser considerado como una herramienta en la toma de decisiones en las áreas de cartografía civil y militar, ordenamiento territorial, agricultura, ingeniería civil y sistemas de información geográfica (SIG), entre otras.

La imagen satelital que conforma el espacio-mapa es utilizada como una referencia espacial introductoria c dad de tener una visión general de la volcán Chichón. Para poder trabaja genes concebidas por medio de percepción remota es necesario darles un tratamiento previo, el cual consiste en la corrección radiométrica y ajuste del histograma. La corrección radiométrica es un procedimiento que resulta necesario cuando se trabaja con niveles digitales de diversas imágenes de la misma área, de diferentes fechas, como un medio para asegurar que los valores de todas las imágenes a trabajar sean equiparables (Chuvieco, 2002). El ajuste de histograma de frecuencia de cada banda se hizo para homogeneizar visualmente la imagen y realizar una primera valoración de la misma, lo que supondrá una mejora del contraste de la imagen.

El espacio-mapa es diseñado a partir de una imagen del satélite SPOT, tomada en 2011 por el sensor 5, con una resolución espacial de 10 m, es decir, el tamaño de lado por pixel. En él se pueden observar rasgos geográficos del noroeste del estado de Chiapas. De manera clara se identifica la estructura volcánica del Chichonal, los cauces de los ríos principales de la región, así como las principales poblaciones de los municipios Francisco León, Chapultenango, Ostuacán, Pichucalco, Ixtacomitán y Sunuapa.

Para elaborar el espacio-mapa se utilizó el compuesto a color con la finalidad de determinar las unidades territoriales, a partir de la mezcla de las bandas verde (1), roja (2) e infrarrojo cercano (3). Este proceso genera una gama de rojos para la vegetación, los más fuertes para la vegetación madura o con un proceso fotosintético mayor (selvas y bosques), los rojos claros corresponden a pastizales y zonas de cultivo de temporal o de riego principalmente, los tonos rosas coinciden con vegetación con un proceso de crecimiento menor y los tonos verdes corresponden a suelo desnudo o seco, producto de procesos antrópicos. Los ríos y las manchas urbanas se distinguen precisamente por su particular tipo de traza (Figura I.4.1).

El uso de las imágenes de satélite en el estudio de la superficie terrestre en cualquiera de sus aplicaciones es de suma importancia y utilidad,

permite una rápida y en ocasiones económica forma de obtener información complementaria del lugar sin estar precisamente en él. Lo anterior, sin olvidar que como cualquier otra técnica, también presenta limitaciones, que pueden ser errores propios del sensor o del lugar como por ejemplo el bandeado de la escena satelital o la nubosidad del lugar al momento de la captura de la imagen.

Figura I.4.1 Vista panorámica del tipo de vegetación que se encuentra en la zona.

Bibliografía

Chuvieco, E. (2002), Teledetección Ambiental. La observación de la Tierra c bacio, Ariel Ciencia, Barcelona. Roset G. Bocco (2003), "Los sistemas de información geográfica y la percepción remota. Herramientas integradas para los planes de manejos en comunidades forestales", Gaceta Ecológica, INE-SEMARNAT, núm. 68, pp. 43-54.

Figura 3.19: Ejemplo de un tema especifico del atlas

En la Figura 3.20 se muestra la interfaz de administración. Para acceder aquí, se debe contar con permisos de superusuario. En esta sección es posible crear, borrar o editar elementos del atlas.

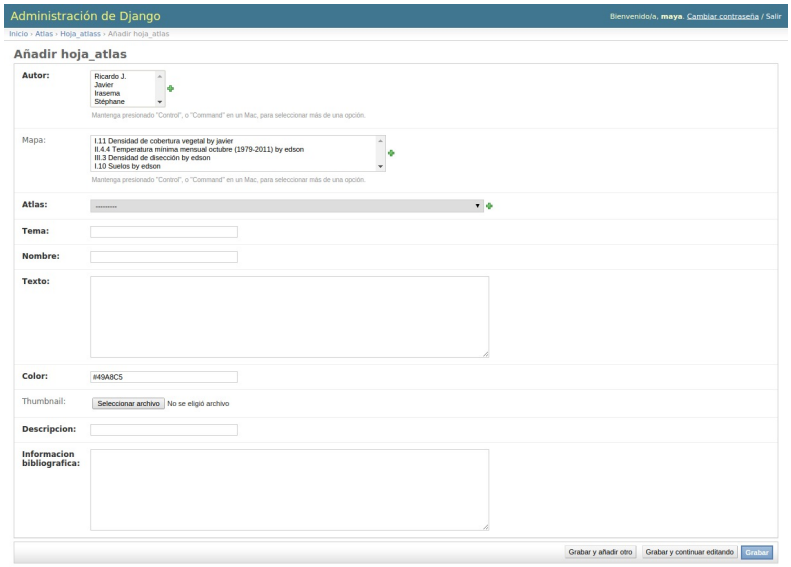


Figura 3.20: Interfaz de administración de django para añadir temas específicos

En la Figura 3.21 se muestra otra sección que también sirve para subir datos al atlas. Se debe contar con permisos para acceder a está parte del sistema.

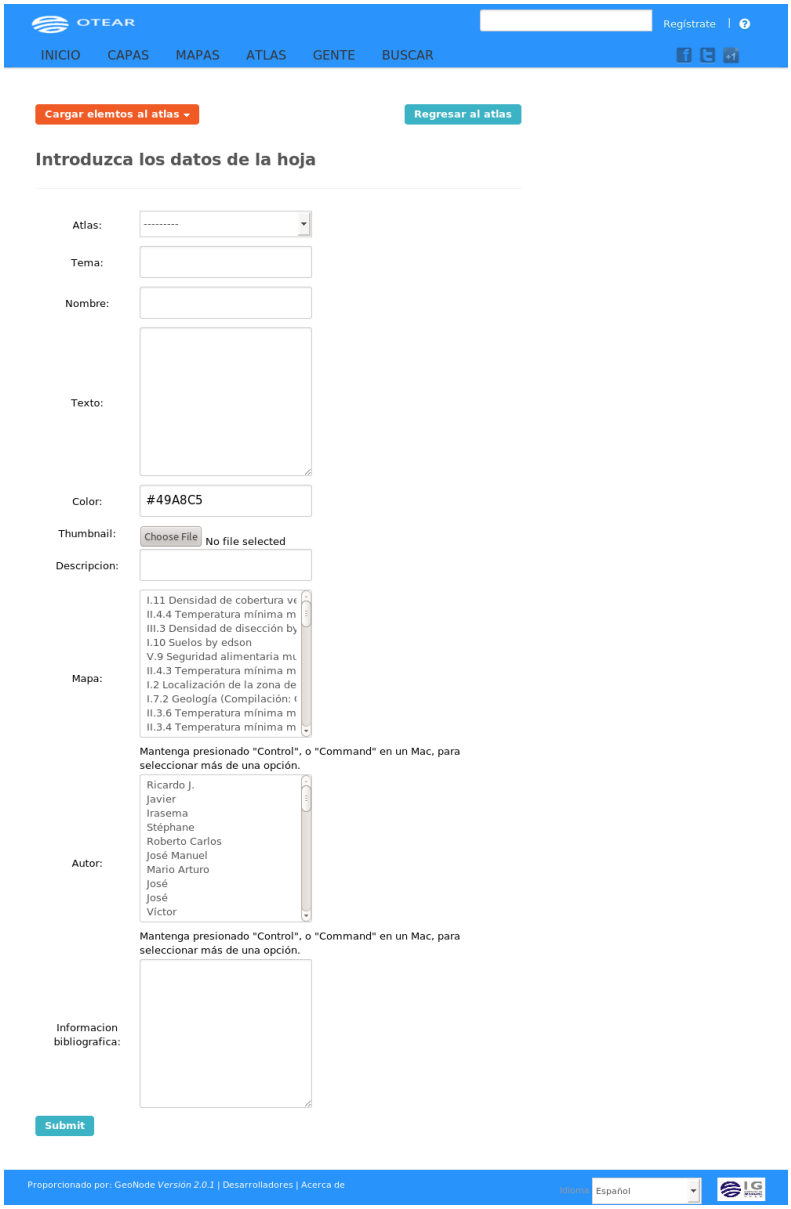


Figura 3.21: Sección para cargar un tema especifico

CAPÍTULO 4

Resultados y Conclusiones

4.1. Introducción

Se implemento una versión digital del atlas *"La región del volcán Chichón, Chiapas: un espacio potencial para su protección, conservación y desarrollo sustentable"*. Esta versión mantiene la calidad de los contenidos(fotos, texto, mapas, gráficas, tablas) y amplia la audiencia de los datos geoespaciales mediante servicios web OGC. Para la construcción del atlas se tomó como referencia la funcionalidad básica de GeoNode y mediante una *app* escrita en Djando se personalizo la estructura u organización para adaptarla a la del tema especifico del atlas impreso. Se decidió extender a este proyecto para aprovechar las características que lo hacen una plataforma geoespacial colaborativa.

En octubre de 2014 se hizo un primer prototipo del atlas. Este solo usaba al componente GeoExt de GeoNode para desplegar los mapas interactivos. La información contenida en los temas específicos no estaba almacenada en la base de datos.

Una vez analizado el potencial del uso de GeoNode, se decidió generar una versión que estuviera debidamente acoplada a GeoNode. Esta haría uso de las herramientas con las que cuenta la plataforma y haría más fácil la administración de los contenidos del atlas.

Se hizo el análisis de requerimientos funcionales y no funcionales para conocer las características que debía tener el atlas. El siguiente paso fue el diseño. Se realizaron diagramas de clases para diseñar la base de datos. Se hicieron prototipos de pantalla a partir de la apariencia de la primer versión para el diseño de la interfaz.

Después en base a los diagramas realizados se implemento el sistema. Se hicieron las tablas de la base de datos de acuerdo al diagrama de clase, se escribieron las funciones

para que el sistema tuviese el comportamiento descrito en el mapa del sitio y que se mostraran los elementos necesarios para cada sección.

Se conecto la *app* a una versión de GeoNode que ya contenía mapas interactivos almacenados y finalmente se copiaron y cargaron todos los temas específicos con sus respectivos elementos.

4.1.1. Problemas encontrados

Cuando se terminó la *app*, se debía instalar para crear el atlas. Hubieron problemas con el proceso de integración debido a que los datos(mapas y capas) y la *app* para GeoNode no eran compatibles por las diferencias en los esquemas de datos. Los datos se encontraban cargados en una instancia con GeoNode 2.0 y la aplicación usaba el esquema de datos para la versión 2.4.

Se pensó en modificar el nuevo esquema de GeoNode para que fuese compatible con el respaldo, pero esto tenía varios inconvenientes, como que esta tarea podría ser complicada y llevaría mucho tiempo o que los cambios podrían no ser compatibles con la versión 2.4. Además de que próximamente se liberará la versión estable[54].

Otra opción fue recuperar los archivos *shapes*, estilos, metadatos y archivos *raster*, pero esto tuvo un problema, el cual era que los mapas no podrían recuperarse, puesto que se encuentran en una tabla de datos e implicaba rehacer los mapas.

Se decidió adecuar la *app* para la versión 2.0. De esta forma solo se modificaría el código del atlas y se dejaría intacta la versión de GeoNode que contiene los datos y se tendrían dos versiones, una estable con la ventaja de no tener que lidiar con la migración hacia una nueva y la otra actualizada, operando bajo GeoNode 2.4.

Un aspecto que requirió trabajo fue conectar la *app* a GeoNode, ya que se tuvieron que realizar cambios debido a las diferentes versiones de Django. La versión de desarrollo usaba la versión 1.7 y la versión 2.0 de GeoNode usa la 1.5.

Para conectar la *app* se encontró en los foros de discusión información que se sirvió para poder instalarla en un ambiente como el que se tenía. El equipo de desarrollo de GeoNode creó una plantilla del proyecto que contiene archivos específicos con toda la configuración de GeoNode que se necesita para conectar una *app* a GeoNode[55]. Esta plantilla sirve más para personalizar la apariencia del sistema como agregar estilos, imágenes o código de JavaScript.

El procedimiento para instalar la *app* fue el siguiente:

1. Descargar la plantilla del repositorio de GitHub.
2. Instalar la plantilla con el instalador de paquetes de python (pip).
3. Reiniciar el servidor para que se ejecutaran los cambios.
4. Crear la *app* con los comandos de *Django*.

5. Copiar el proyecto a la aplicación recién creada.

6. Agregar la aplicación a la lista de aplicaciones instaladas en GeoNode.

7. Sincronizar la base de datos para que se leyera el esquema de la aplicación y se crearan las tablas del atlas en la base de datos de GeoNode.

4.1.2. Resultados

El atlas se encuentra instalado en un servidor de prueba para realizar mejoras y corregir errores del sistema. La versión en la que se encuentra instalado el atlas es la 2.0. Se espera que el sitio se publique para que sea accesible desde la página del instituto de geografía.

4.1.3. Conclusiones

Después de haber aprendido a usar Django, conocer más detalles sobre la arquitectura y cualidades de GeoNode e implementar el atlas se concluye que fue una buena elección por las siguientes razones:

- Los modelos en Django son clases que proveen orientación a objetos. Esto facilita la creación de los formularios, vistas y sección de administración. En cada caso solo es necesario escribir una o dos lineas de código para cada componente.
- Django cuenta con vistas genéricas que ayudan a mostrar los atributos o detalles de un objeto. Esto evita que el programador cree el mismo tipo de vistas en cada proyecto.
- Con el sistema de plantillas Django separa la lógica de la vista de una página. Esto facilita modificar la apariencia sin alterar el código de Python subyacente.
- Django se encarga de abstraer la mayor parte del trabajo con formularios como es la validación de los datos o la presentación al usuario.
- Para este atlas resultó ser apropiada la decisión de extender a GeoNode ya que al estar construido con Django permite construir módulos independientes y relativamente fáciles de conectar.
- GeoNode cuenta con una extensa comunidad de programadores y usuarios que se encargan de reportar errores, crear documentación, ayudar a responder dudas en los foros y la programación del sistema.
- Django está basado en la arquitectura de diseño MVT, lo que hace fácil identificar la función de cada archivo.
- La configuración del *framework* es sencilla y se hace en un solo archivo (settings.py).

Las desventajas que se encontraron son:

- Hacer consultas más complejas como *joins* a la base de datos no es tan fácil, debido a que estas no se hacen directamente en código SQL, sino en código de python usando funciones que proporcional el ORM.
- La diferencia entre las bibliotecas que necesita GeoNode y las que trae la versión del sistema operativo por defecto, suele provocar errores.
- La documentación de GeoNode no es precisa. Esto se debe a que al actualizar el repositorio de GitHib con código nuevo, pero no se incluyen instrucciones actualizadas para hacerlo funcionar.

Respecto a la dos versiones en las que se desarrolló el presente atlas, la primera sirvió como prototipo para desarrollar el sistema en Django, resultó ser más sencillo de mantener que la versión desarrollada con HTML estáticos, debido a que utiliza la infraestructura de software que brinda Django y la arquitectura MVT, los cuales brindan una mejor organización de los componentes del sistema, obteniendo una manera sencilla de identificar cada componente, así como las partes que conforman cada uno de éstos, además de que Django evita la inyección de código SQL gracias a los *Managers* que son los encargados de realizar las operaciones en la base de datos para cada objeto. Esta característica es muy útil porque brinda mayor seguridad al sistema. Aunque en ocasiones una consulta a la base de datos se vuelve más compleja.

Una una herramienta de este tipo puede ser de mucha utilidad para publicar material que contenga datos geográficos digitales y que más personas accedan mas fácilmente a este.

En conclusión, el atlas en línea cumple con su propósito, el cual es generar una versión en línea del atlas. Por medio de este es posible almacenar la información contenida en el atlas en una base de datos, se puede acceder y manipular información geográfica del área del volcán mediante los servicios OGC que implementa GeoServer.

Para finalizar, espero que mi herramienta contribuya a facilitar la comunicación y colaboración entre las autoridades de protección civil, la comunidad científica involucrada y las autoridades para impulsar impulsar la creación de un geoparque en la zona con el objetivo triple de mejorar la economía de los habitantes de la región, proteger el patrimonio geológico y minimizar los riesgos de desastre a través de la gestión activa e informada de los riesgos.

4.1.4. Trabajo a futuro

Para mejorarlo se podría añadir un modo de edición para los datos de los temas específicos. Cambiar las imágenes de las leyendas por la leyenda que genera el componente de GeoExt para interactuar de mejor manera con las capas del mapa. También revisar el esquema de la base de datos o renovarlo en caso de que se necesario ajustarlo a una nuevo

esquema de datos de GeoNode. Para las siguientes actualizaciones de GeoNode hacer las adecuaciones para que el atlas pueda seguir funcionando adecuadamente con GeoNode y sus demás componentes.

Otro tarea a futuro es actualizar el sistema a la versión 2.4 que se encuentra liberada como estable. Se espera poder contar con los datos necesarios para generar los mapas interactivos que faltan y reemplazar a las imágenes de mapas.

Anexos

ANEXOS A

Anexo I: Instalación de GeoNode para el desarrollo

En el sitio de internet de GeoNode existe una sección destinada a los desarrolladores [1], en ésta se encuentran instrucciones a seguir para instalar el ambiente de desarrollo. Se debe mencionar que para cada versión de GeoNode el entorno de desarrollo cambia debido a las versiones de los paquetes o a nuevas adiciones de bibliotecas a GeoNode. En este caso se utilizo la versión 2.4.dev. La versión 2.4 brinda instrucciones para instalarse en CentOS7 [2] y en Windows[3].

Los pasos son los siguientes:

1. Actualizar la lista de paquetes del sistema operativo.

   ```
   $ sudo apt-get update
   ```

2. Instalar herramientas de construcción y bibliotecas.

   ```
   $ sudo apt-get install -y build-essential libxml2-dev
     libxslt1-dev libpq-dev zlib1g-dev
   ```

3. Instalar dependencias nativas de *Python*

   ```
   $ sudo apt-get install -y python-dev python-imaging python-
     lxml python-pyproj python-shapely python-nose python-
     httplib2 python-pip python-software-properties
   ```

   Instalar el ambiente virtual de Python(Python Virtual Environment)

[1] http://geonode.readthedocs.org/en/latest/tutorials/devel/index.html
[2] http://docs.geonode.org/en/master/tutorials/installandadmin/setuponcentos/setupcentos.html
[3] http://docs.geonode.org/en/master/tutorials/installandadmin/quickinstall.html

```
$ sudo pip install virtualenvwrapper
```

Postgresql

```
$ sudo apt-get install postgresql-9.3-postgis-2.1 postgresql
    -9.3-postgis-scripts
```

Cambiar la contraseña de *Postgres*

```
$ sudo passwd -u postgres
$ sudo passwd postgres
```

Crear el *role* geonode y la base de datos.

```
$ su postgres
$ createdb geonode
$ psql
postgres=#
postgres=# \password postgres
postgres=# CREATE USER geonode WITH PASSWORD 'geonode';
# should be same as password in setting.py
postgres=# GRANT ALL PRIVILEGES ON DATABASE "geonode" to
    geonode;
postgres=# \q
```

Dependencias de *Java*

```
$ sudo apt-get install -y --force-yes openjdk-6-jdk ant
    maven2 --no-install-recommends
```

Herramientas de apoyo

```
$ sudo apt-get install -y git gettext
```

4. Establecer un entorno virtual. Aquí es donde correrá GeoNode.
   Agregar el *virtualenvwrapper* al nuevo ambiente

```
$ export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python
$ export WORKON_HOME=~/.venvs
$ source /usr/local/bin/virtualenvwrapper.sh
$ export PIP_DOWNLOAD_CACHE=$HOME/.pip-downloads
```

   Configurar el entorno virtual local para GeoNode

```
$ mkvirtualenv geonode
$ workon geonode
```

5. En este paso comienza la instalación de GeoNode. El primer paso es descargar el código de la última versión desde *GitHub*, se utiliza el comando *clone*

```
$ git clone https://github.com/GeoNode/geonode.git
```

6. Agregar *nodejs* al PPA(Personal Package Archive) y otras herramientas necesarias para la creación de las vistas.

```
$ sudo add-apt-repository -y ppa:chris-lea/node.js
$ sudo apt-get update
$ sudo apt-get install -y nodejs
$ cd geonode/static
$ npm install --save-dev

# En caso de que el ultimo comando no funcione, se pueden
    ejecutar los siguientes comandos:

$ npm install bower --save-dev
$ npm install grunt-cli --save-dev
$ npm install grunt-contrib-jshint --save-dev
$ npm install grunt-contrib-less --save-dev
$ npm install grunt-contrib-concat --save-dev
$ npm install grunt-contrib-copy --save-dev
$ npm install grunt-text-replace --save-dev
$ npm install grunt-contrib-uglify --save-dev
$ npm install grunt-contrib-cssmin --save-dev
$ npm install grunt-contrib-watch --save-dev
```

7. Instalar GeoNode en el nuevo ambiente virtual

```
$ pip install -e geonode --use-mirrors
$ cd geonode
```

8. Compilar e iniciar el servidor

```
$ paver setup
```

9. Iniciar nuestra instancia de GeoNode

```
$ paver start
```

Podemos visitar nuestra versión de GeoNode en la dirección http://localhost:8000 usando cualquier navegador.

10. Crear un superusuario para *django* y *geonode*

```
$ django-admin.py createsuperuser --settings=geonode.
    settings
```

Bibliografía

[1] D. la Cruz-Reyna, A. Martin Del Pozzo, *et al.*, "The 1982 eruption of el chichón volcano, méxico: Eyewitness of the disaster," *Geofísica internacional*, vol. 48, no. 1, pp. 21–31, 2009.

[2] Proceso, "El chichonal, 25 años de la tragedia." `http://hemeroteca.proceso.com.mx/?p=206647`, 2015. [Web; consultado septiembre 2015].

[3] I. Alcántara, R. J. Garnica, A. Coll-Hurtado, and S. G. Ramos, *La región del volcán Chichón, Chiapas: un espacio potencial para su protección, conservación y desarrollo sustentable.* UNAM, 2013.

[4] J. Osorno-Covarrubias *et al.*, "An online atlas as a collaborative and visualization tool for the geopark proposal of the chichonal volcano area," in *XV Simposio Internacional del SELPER, Medellín, Colombia.*, p. 535, 2013 septiembre.

[5] G. Giuliani, P. Lacroix, Y. Guigoz, L. Bigagli, N. Ray, and A. Lehmann, *Bringing GEOSS services into practice.* Gregory Giuliani, 2014.

[6] D. Fraser Taylor and S. Caquard, "Cybercartography: maps and mapping in the information era," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 41, no. 1, pp. 1–6, 2006.

[7] ESRI, "Geographical feature." `http://support.esri.com/en/knowledgebase/GISDictionary/term/feature`, 2015. [Web; consultado julio 2015].

[8] K.-J. L. C. Vangenot, "Web and wireless geographical information systems," p. 45, 2005.

[9] ESRI, "Gis dictionary." `http://support.esri.com/en/knowledgebase/GISDictionary/term/map`, 2015. [Web; consultado mayo 2015].

[10] ESRI, "Cómo transmiten los mapas la información geográfica." http://resources.arcgis.com/es/help/getting-started/articles/026n0000000q000000.html, 2015. [Web; consultado junio 2015].

[11] A. Aitchison, *Beginning spatial with SQL Server 2008*. Apress, 2009.

[12] R. Kothuri, A. Godfrind, and E. Beinat, *Pro oracle spatial for oracle database 11g*. Dreamtech Press, 2008.

[13] A. K. Jha and J. E. Duyne, *Safer homes, stronger communities: a handbook for reconstructing after natural disasters*. World Bank Publications, 2010.

[14] N. L. Zarzosa and M. A. N. Andrés, *Sistemas de información geográfica. Prácticas con Arc View*, vol. 120. Univ. Politéc. de Catalunya, 2004.

[15] O. G. Consortium, "Web processing service implementation specification." http://portal.opengeospatial.org/files/?artifact_id=31137. [Web; consultado junio 2015].

[16] L. A. Nolan, P. G. Andrew, and M. Bidney, "The digital atlas dilemma: Outlining the challenges for libraries," *Journal of Map & Geography Libraries*, vol. 10, no. 2, pp. 134–135, 2014.

[17] M. Minghini, *Multi-dimensional GeoWeb platforms for citizen science and civic engagement applications*. PhD thesis, Italy, 2014.

[18] L. Bermúdez, "Interoperability and the value of standards," *Realidad, Datos y Espacio Revista Internacional de Estadística y Geografía*, pp. 103–110, 2009.

[19] O. G. Consortium, "Welcome to the ogc." http://www.opengeospatial.org/, 2015. [Web; consultado octubre 2015].

[20] O. G. Consortium, "About ogc." http://www.opengeospatial.org/ogc, 2015. [Web; consultado octubre 2015].

[21] ESRI, "Policies." http://www.opengeospatial.org/ogc/policies, 2016. [Web; consultado Enero 2016].

[22] IBM, "Servicio web." http://www.ibm.com/developerworks/ssa/webservices/newto/service.html, 2015. [Web; consultado diciembre 2015].

[23] O. G. Consortium, "Web map server implementation specification," 2015. [Web; consultado agosto 2015].

[24] O. G. Consortium, "Geography markup language." http://www.opengeospatial.org/standards/gml, 2015. [Web; consultado junio 2015].

[25] O. G. Consortium, "Web feature service implementation specification," 2015. [Web; consultado mayo 2015].

[26] O. G. Consortium, "Web map tile service implementation specification." https://portal.opengeospatial.org/files/?artifact_id=35326, 2010. [Web; consultado junio 2015].

[27] O. G. Consortium, "Web coverage service implementation specification." https://portal.opengeospatial.org/files/09-110r4. [Web; consultado junio 2015].

[28] O. G. Consortium, "Web processing service implementation specification." http://portal.opengeospatial.org/files/?artifact_id=24151. [Web; consultado junio 2015].

[29] Wikipedia, "Rich internet application." https://en.wikipedia.org/wiki/Rich_Internet_application, 2015. [Web; consultado diciembre 2015].

[30] OpenLayers, "Openlayers 3." http;//openlayers.org, 2015. [Web; consultado febrero 2015].

[31] OpenStreetMap, "Openstreetmap." view-source:https://www.openstreetmap.org/about, 2015. [Web; consultado agosto 2015].

[32] MapQuest, "Mapquest." http://company.mapquest.com/, 2015. [Web; consultado agosto 2015].

[33] Pycsw, "Metadata publishing just got easier." http://pycsw.org/, 2015. [Web; consultado marzo 2015].

[34] OsGeo, "Guía de inicio rápido de los servicios web deegree 3." http://live.osgeo.org/es/quickstart/deegree_quickstart.html). [Web; consultado febrero 2015].

[35] T. O'reilly, "What is web 2.0: Design patterns and business models for the next generation of software," *Communications & strategies*, p. 17, 2007.

[36] GeoNode, "What's this geonode thing?." http://digifesto.com/2010/02/18/whats-this-geonode-thing/, Julio 2015. [Web; consultado junio 2015].

[37] D. C. Schmidt, "Applying patterns and frameworks to develop object-oriented communication software," *Handbook of programming languages*, vol. 1, 1997.

[38] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[39] J. Bucanek, *Model-View-Controller Pattern*. Springer, 2009.

[40] python.org, "Welcome to python.org." https://www.python.org. [Web; consultado febrero 2016].

[41] python.org, "Applications for python." https://www.python.org/about/apps/. [Web; consultado febrero 2016].

[42] Django, "jango makes it easier to build better web apps more quickly and with less code.." https://www.djangoproject.com/, Diciembre 2015. [Web; consultado diciembre 2015].

[43] python.org, "Django was invented to meet fast-moving newsroom deadline.." https://www.djangoproject.com/start/overview/. [Web; consultado febrero 2016].

[44] GeoNode, "Django apps." http://geonode.readthedocs.org/en/latest/reference/developers/django-apps.html, 2015. [Web; consultado junio 2015].

[45] Microsoft, "¿qué es un servidor proxy?." http://windows.microsoft.com/es-mx/windows-vista/what-is-a-proxy-server, 2015. [Web; consultado agosto 2015].

[46] GeoNode, "Components." http://geonode.readthedocs.org/en/latest/reference/components.html, 2015. [Web; consultado junio 2015].

[47] OpenGeo, "Geoexplorer." http://suite.opengeo.org/opengeo-docs/geoexplorer/, 2015. [Web; consultado mayo 2015].

[48] GeoNode, "pycsw." http://geonode.readthedocs.org/en/latest/reference/components.html#pycsw, 2015. [Web; consultado junio 2015].

[49] jQuery, "What is jquery?." https://jquery.com/, 2015. [Web; consultado agosto 2015].

[50] w3schools, "Bootstrap get started." http://www.w3schools.com/bootstrap/bootstrap_get_started.asp, 2015. [Web; consultado agosto 2015].

[51] Sencha, "Sencha ext js." https://www.sencha.com/products/extjs/#overview, 2015. [Web; consultado septiembre 2015].

[52] GeoExt2, "Geoext 2 — javascript toolkit for rich web mapping applications." https://geoext.github.io/geoext2/, 2015. [Web; consultado mayo 2015].

[53] V. Fernández Alarcón, *Desarrollo de sistemas de información: Una metodología basada en el modelado*. Edicions UPC, 2006.

[54] T. G. dev team, "Geonode devel - geonode 2.4 status." http://osgeo-org.1560.x6.nabble.com/Geonode-2-4-status-td5192355.html, 2015. [Web; consultado julio 2015].

[55] S. Dalmaso, "Zinnia blog app in a geonode installed with apt-get." https://groups.google.com/forum/#!topic/geonode-users/9CfJWD7X6Jk, 2015. [Web; consultado julio 2015].